Centro Federal de Educaçào Tecnológica de Minas Gerais
Campus Divinópolis
Graduação em Engenharia Mecatrônica

Matheus Martins Coelho

Módulo didático para aprendizado de programação e robótica

Divinópolis
2022

Matheus Martins Coelho

Módulo didático para aprendizado de programação e robótica

Monografia de Trabalho de Conclusão de Curso apresentada ao Colegiado de Graduação em Engenharia Mecatrônica como parte dos requisitos exigidos para a obtenção do título de Engenheiro Mecatrônico.

Eixo de Formação: Computação e Eletrônica.

Orientador: Prof. Dr. Alan Mendes Marotta

Divinópolis

2022

Matheus Martins Coelho

Módulo didático para aprendizado de programação e robótica

Monografia de Trabalho de Conclusão de Curso apresentada ao Colegiado de Graduação em Engenharia Mecatrônica como parte dos requisitos exigidos para a obtenção do título de Engenheiro Mecatrônico.
Eixo de Formação: Computação e Eletrônica.

Comissão Avaliadora:

Prof. Dr. Alan Mendes Marotta
CEFET-MG / *Campus* Divinópolis

Prof. Me. Marlon Henrique Teixeira
CEFET-MG / *Campus* Divinópolis

Prof. Dr. Thiago Magela Rodrigues Silva
CEFET-MG / *Campus* Divinópolis

Divinópolis
2022

A minha família que sempre me apoiaram durante toda trajetória acadêmica.

Agradecimentos

Agradeço,

primeiramente a Deus por sempre estar presente em minha vida com suas bençãos diárias;

Aos meu pais, Dante e Ilza e minha irmã Caroline por todo incentivo, amor e esforços realizados para me ajudar na conclusão de mais essa etapa em minha vida. Vocês são peças essenciais para mim, amo vocês!

A Maria Clara que sempre acreditou na minha capacidade e acompanhou de perto todo o desenvolvimento deste trabalho, além de sempre estar ao meu lado me dando forças;

Aos meus amigos e companheiros de curso, em especial a Turma 8, por toda convivência diária sempre com muito companheirismo e alegria, mesmo nos momentos mais difíceis do curso;

Ao Prof. Alan por toda paciência, esclarecimentos e contribuições nesse período de orientação;

Aos professores e funcionários do CEFET-MG Campus Divinópolis, pela extrema dedicação, colaboração e atenção;

Enfim, a todos os que por algum motivo contribuíram em minha trajetória acadêmica.

A melhor aprendizagem virá das melhores oportunidades de construir.

Seymour Papert

Resumo

Recentemente, surgiu-se muitas escolas e propostas para o ensino de programação, tal movimentação se deu após a demonstração empírica dos vários benefícios que este ensino pode proporcionar. Além disso, muitas tecnologias já estão inseridas em diversos setores da sociedade e diante do fato da educação básica ter como objetivo preparar o cidadão para o mundo, é fundamental que os conceitos e práticas desta área sejam incluídos nas redes escolares de todo Brasil, como já tem sido feito por vários países. Somado a isto, temos dentre as 10 competências da Base Nacional Comum Curricular (BNCC) a "Cultura Digital": compreender, utilizar e criar tecnologias digitais de forma crítica, significativa e ética. Em contrapartida, nos esbarramos com o alto valor financeiro dos kits didáticos de robótica e a necessidade do uso de um computador para programá-los, fato que dificulta a aplicação deste ensino em escolas públicas e a prática dos alunos em suas casas. Diante desta realidade este trabalho tem como objetivo desenvolver um kit didático de baixo custo para um ensino e prática de programação mais acessíveis. Para isto, é proposto neste trabalho o desenvolvimento de um aplicativo, que possuirá em sua primeira versão, um banco de conteúdos para o aprendizado de programação, robótica e eletrônica básica. A segunda versão com o editor de códigos e a última com a implementação da conexão sem fio com o microcontrolador para a gravação do mesmo. Este trabalho se propôs a criar a primeira versão do aplicativo. Propôs também o desenvolvimento de um dispositivo com sensores e atuadores integrados ao microcontrolador, para a prática da programação e robótica.

Palavras-chave: Robótica Educacional, Dispositivo móvel, Módulo Didático, Ensino de Programação.

Abstract

Recently, came up many schools and proposals for teaching programming, this movement took place after the empirical demonstration of the many benefits that this teaching can provide. In addition, many technologies are already inserted in different sectors of society and given the fact that basic education aims to prepare citizens for the world, it is essential that the concepts and practices of this area are included in school networks throughout Brazil, as already has been done by several countries. Added to this, we have among the 10 competencies of the BNCC the "Digital Culture": understanding, using and creating digital technologies in a critical, meaningful and ethical way. On the other hand, we are faced with the high financial value of robotics kits and the necessity to use a computer to program them, a fact that makes it difficult to apply this teaching in public schools and the practice of students at home. Given this reality, this work aims to develop a low-cost teaching kit for more accessible teaching and programming practice. For this, proposes the development of an application, which will have, in its first version, a content bank for learning programming, robotics and basic electronics. The second version with the code editor and the last one with the wireless connection with the microcontroller for recording the same. This work proposes to create the first version of the application. Also proposes the development of a device with sensors and actuators integrated with the microcontroller, for the practice of programming and robotics.

Keywords: Educational Robotics, Mobile Device, Didactic Module, Programming Teaching.

Sumário

Lista de figuras

Lista de tabelas

Lista de acrônimos e notações

ABNT	Associação Brasileira de Normas Técnicas
BNCC	Base Nacional Comum Curricular
TIC	Tecnologia da Informação e Comunicação
STEAM	Ciência, Tecnologia, Engenharia, Artes e Matemática, do inglês *Science, Technology, Engineering, Arts and Mathematics*
SBC	Sociedade Brasileira de Computação
CEIE	Comissão Especial de Informática na Educação
SBIE	Simpósio Brasileiro de Informática e Educação
RBIE	Revista Brasileira de Informática na Educação
IoT	Internet das Coisas, do inglês *Internet of Things*
LED	Diodo Emissor de Luz, do inglês *Light-Emitting Diode*
MIT	Instituto de Tecnologia de Massachusetts, do inglês *Massachusetts Institute of Technology*
LDR	Resistor Dependente de Luz, do inglês *Light Dependent Resistor*
LDB	Aprender fazendo, do inglês *Learning By Doing*
CAI	Instrução auxiliada por computador, do inglês *Computer-Aided Instruction*
PEC	Programas Educacionais por Computador

PWM Modulação por largura de pulso, do inglês *Pulse Width Modulation*

GPIO Entrada/Saída de Uso Geral, do inglês *General Purpose Input/Output*

IR Receptor Infravermelho

AD Analógico Digital

Capítulo 1

Introdução

No contexto atual, é de comum acordo entre cientistas da computação e profissionais da educação que a tecnologia deve estar inserida na experiência educacional dos alunos. Segundo uma pesquisa da Google em conjunto com a Gallup, feita em escolas americanas do ensino fundamental e médio, 84% dos pais, 71% dos professores e 64% dos diretores dizem que oferecer ciências da computação nas escolas é mais importante ou tão importante quanto cursos obrigatórios como matemática, ciências, história e inglês (GOOGLE, 2017).

E não somente levar a tecnologia para dentro da sala de aula como mera exposição, segundo Machado et al. (2016), a tecnologia deve levar o aluno a ser um pensador criativo, capaz de se expressar e mostrar suas ideias ao mundo, conferindo-lhe cada vez mais autonomia para atuar na sociedade da informação digital (MACHADO et al., 2016) .

Em 1980 já se tinha experiências desenvolvidas na Educação Básica fora do Brasil, utilizando a linguagem de programação LOGO como ambiente de exploração, iniciativas estas baseadas na proposta do professor do MIT, Seymour Papert. Segundo Papert (1980), no passado, ao entrar na casa de crianças observaríamos que as ferramentas utilizadas para estudo eram os livros, cadernos e lápis. Atualmente, além destes materiais, as crianças também utilizam smartphones, tablets, internet, redes sociais e outros meios digitais. Portanto, as escolas devem acompanhar esta evolução, utilizar estas ferramentas para o ensino e, sobretudo, instruir a como utilizar estas ferramentas tecnológicas, visando o desenvolvimento de diferentes habilidades.

As tecnologias devem ser utilizadas em contextos de formação e desenvolvimento, elas geram crescimento pessoal e influenciam positivamente no futuro daqueles que as domi-

nam e utilizam adequadamente. Para o uso pertinente e responsável das tecnologias, é necessário explorá-las nos diversos contextos educacionais. Se faz importante a contribuição não somente para acabar com o analfabetismo tecnológico, mas, principalmente, para que as tecnologias façam parte da aprendizagem de crianças e adolescentes, favorecendo e ampliando suas perspectivas em atuações futuras (CODE8734, 2021).

A atividade de programar poderia ser adicionada às competências básicas de leitura e escrita, pois auxilia no desenvolvimento cognitivo, permitindo que crianças e adolescentes vejam o mundo de maneiras diferentes (CODE8734, 2021). Segundo Resnick (2013), "[...] a programação contribui não apenas para aprender a resolver problemas, mas em transformar ideias por meio da criação, da exploração e da experimentação". Além disso, a prática de programação também pode contribuir para a melhora na assimilação de conceitos abstratos, da escrita, raciocínio lógico, raciocínio matemático e desenvolve habilidades de planejamento e organização.

A fluência tecnológica é um assunto recorrente no cenário atual. O fluente tecnológico é aquele capaz de otimizar sua interação com a tecnologia, sabendo usar os recursos de forma intuitiva, criativa, consciente e colaborativa.

Em relação ao contexto histórico do ensino de programação e robótica para crianças e adolescentes, o Reino Unido foi o primeiro país, em 2014, a tornar obrigatório este ensino no ensino básico, incluindo a disciplina em currículo nacional para alunos a partir de 5 anos de idade. Com isso, os países líderes no ranking mundial de educação, logo compreenderam a importância e adotaram a mesma postura referente ao ensino de programação no ensino básico, tornando-a obrigatória na grade curricular.

Recentemente, o Brasil viu a homologação da BNCC para a Educação Básica - Educação Infantil e Ensino Fundamental (2017) e Ensino Médio (2018). A premissa fundamental é que a educação caminhe na direção estabelecida pelo artigo 205 da Constituição Federal: "pleno desenvolvimento da pessoa, seu preparo para o exercício da cidadania e sua qualificação para o trabalho". Das dez competências gerais da BNCC para a Educação Básica, a 5ª competência refere-se à "Cultura Digital" – "compreender, utilizar e criar tecnologias digitais de forma crítica, significativa e ética" (SIQUEIRA, 2019).

Porém, ao observar a realidade brasileira nos esbarramos com o alto valor agregado dos kits didáticos de robótica e na necessidade do uso de um computador para programar. Fato que dificulta a aplicação deste ensino em escolas públicas e a prática dos alunos

em suas casas. Afinal, segundo os dados da pesquisa "Tecnologia da Informação e Comunicação – TIC" do IBGE em 2019, apenas 40,6% dos domicílios brasileiros possuíam um computador, por outro lado, a parcela das residências em que havia aparelho celular alcançou 94% (IBGE, 2019).

Diante desta realidade este trabalho tem como objetivo desenvolver um módulo didático para um ensino e prática de programação mais acessíveis. Para isto, propõe-se o desenvolvimento de um aplicativo e um módulo com sensores e atuadores integrados ao microcontrolador, para a prática de programação e robótica. O aplicativo em questão possuirá em sua primeira versão, um banco de conteúdos para o aprendizado de programação, robótica e eletrônica básica. Uma segunda versão com o editor de códigos e a última com a implementação da conexão sem fio (*Wi-fi*) com o microcontrolador para a gravação do mesmo. Este trabalho se propõe a criar a primeira versão do aplicativo.

1.1 Definição do Problema

Como descrito na seção anterior, o ensino e prática de programação ainda são inacessíveis para grande parte da população brasileira. Muito desse fato se dá pelo alto valor agregado dos kits de robótica e computadores, necessários para a criação dos códigos. Sabendo disso, percebeu-se a necessidade da criação de uma alternativa para viabilizar a prática de programação em escolas públicas e nas casas dos alunos. Com isso, observou-se que um aplicativo que permitisse o aprendizado e a prática de programação apenas com o uso de celulares tende a solucionar grande parte do problema.

1.2 Motivação

Este trabalho tem como motivação tornar mais acessível o ensino e prática de programação, não somente nas escolas brasileiras, mas também dentro das casas dos alunos. Afinal como citado anteriormente neste documento, há uma dificuldade na aquisição e utilização de um computador. E para usarmos uma alternativa *mobile* para gravação do microcontrolador, o aparelho deve ter compatibilidade com USB-OTG (*On The Go*), para que possa se comportar como controlador (*hosts*) de outros aparelhos, fato que não está presente em todos os celulares.

Dito isto, a comunicação sem fio pode ser uma forte alternativa para termos um avanço, para, além do uso de computadores, o uso de celulares e *tablets* para que as pessoas tenham contato com a programação em linha de código, primeiramente com uma versão inicial do aplicativo.

1.3 Objetivos do trabalho

São objetivos do trabalho a serem desenvolvidos:

1.3.1 Objetivos Gerais

Este trabalho tem como objetivo desenvolver um aplicativo em sua primeira versão, que possuirá um banco de conteúdos sobre programação, eletrônica e robótica. Além disso, propõe-se o desenvolvimento de uma placa com sensores e atuadores integrados para ser utilizada no aprendizado e prática de programação e robótica.

1.3.2 Objetivos Específicos

- Estudar e compreender a realidade brasileira no ensino de programação para a educação básica;
- Pesquisar e escolher os componentes adequados para o dispositivo;
- Projetar e desenvolver o circuito de alimentação para o dispositivo;
- Dimensionar, projetar e desenvolver o dispositivo;
- Planejar os requisitos de *software* necessários ao desenvolvimento desta aplicação;
- Estudar sobre a comunicação sem fio entre o aplicativo e o microcontrolador;
- Criar esboço de layout das telas do aplicativo;
- Desenvolver a primeira versão do aplicativo, contanto com a tela de menu, área com banco de conteúdos sobre programação e outra sobre eletrônica básica e robótica;

1.4 Estado da Arte

Sabe-se que um dos papéis da educação é contribuir para a formação dos alunos e prepará-los para os desafios do futuro. Em tempos atuais, não é possível colocar isso em prática sem pensar na atuação desses estudantes frente ao mundo tecnológico.

A sociedade está envolvida por recursos tecnológicos, e fazer uso deles se tornou uma necessidade. Não basta, entretanto, lidar de forma passiva com tecnologia. Precisamos valer-se da tecnologia com fluência, que não implica apenas em ser capaz de utilizar recursos tecnológicos, mas também construir algo com significado a partir de suas próprias ideias e desenvolver diversas habilidades e competências com base na prática de programação (CODE8734, 2021). Afinal, segundo Tim Cook, CEO da Apple: "A programação deveria ser uma segunda língua, ensinada a todas as crianças".

Neste contexto, a inserção de recursos computacionais e tecnológicos na educação tem provocado uma verdadeira revolução na percepção de muitos sobre ensino e aprendizagem (VALENTE, 1998). Muitas metodologias de ensino surgiram diante deste novo contexto, a exemplo temos as metodologias ativas, em que o aluno se torna protagonista do processo de aprendizagem, como a metodologia STEAM (Ciência, Tecnologia, Engenharia, Artes e Matemática), Construcionismo, *Learning By Doing* (LDB), Cultura *Maker*, entre outras.

O ensino com o uso de tecnologias tem suas origens no ensino através das máquinas. Tudo começou com Dr. Sidney Pressey em 1924, quando inventou uma máquina capaz de corrigir testes de múltipla escolha. Em meados de 1950, o professor de Harvard B.F.Skinner adaptou a ideia de Pressey aplicando uma máquina para ensinar usando o conceito de instrução programada, que consistia em módulos que seriam liberados após a realização a aprovação em pequenos testes, se aproximando de uma dinâmica de jogos e níveis (VALENTE, 1998).

A partir dos anos 60 e com o advento do computador, programas de instrução programada foram implementados nestas novas máquinas e, assim, surge a instrução auxiliada por computador, também conhecida como CAI (*computer-aided instruction*) e no Brasil como PEC (Programas Educacionais por Computador)(VALENTE, 1998).

Um grande avanço no ensino com recursos tecnológicos se deu em 1964, quando o professor do MIT (*Massachusetts Institute of Tecnology*) Seymour Papert criou a linguagem LOGO. O professor foi um dos pioneiros no ensino de programação e robótica como temos atualmente. Ele inicialmente utilizava uma plataforma robótica móvel programada por

computador e na sequencia também utilizou uma personagem gráfica, uma tartaruga que poderia ser controlada através da programação. Mais tarde, foi desenvolvido um kit que contava com uma interfase de programação, motores, sensores e peças da marca LEGO (KATO; BRAGA; PAZMINO, 2015).

Nos anos 1980, com a popularização dos microcontroladores ampliou-se a utilização de recursos tecnológicos na educação. No Brasil, no final dos anos 80, foi criada na SBC (Sociedade Brasileira de Computação) a CEIE (Comissão Especial de Informática na Educação) que contribuiu para o começo da utilização de computadores nas escolas. Mas foi apenas por volta de 2010 que inicia uma movimentação para o ensino de programação e robótica, porém, atualmente ainda não aplicado em escolas públicas e em algumas escolas particulares. Diferentemente dos países europeus que desde 2014 já tem este ensino como obrigatório em todas as escolas (CASTRO; GIMENES; C. FILHO, 2019).

Ao redor do mundo existem diversos projetos de kits, softwares e propostas de ensino de programação e robótica, sendo este ensino popularizado com o nome de Robótica Educacional ou Robótica Pedagógica. Dentre os projetos pode-se citar o desenvolvido pelas Universidades de Columbia e Cambridge no verão de 2003 com crianças. O objetivo desta proposta era aumentar o conhecimento de ciências e matemática através do uso do kit LEGO *Mindstormms Robotics Invention System* (GOLDMAN; EGUCHI; SKLAR, 2004).

Outro projeto aplicado fora do Brasil é o *Educational Robotics inSpecial Education*, que consiste em um projeto voltado para a educação de crianças com deficiências com o uso de recursos tecnológicos (LIN et al., 2006). A NASA também possui seu projeto para ensinar programação e robótica, voltado para estudantes do ensino médio a proposta consiste em fomentar o número de alunos interessados nas áreas de ciências e tecnologias.

No Brasil não temos a presença forte de empresas desenvolvedoras de kits de robótica e *softwares* para o ensino de programação, mas tem-se muitos estudos e pesquisas envolvendo metodologias e propostas de ensino com uso de kits já disponíveis no mercado. Além disso, recentemente tem-se a presença de alguns projetos de kits utilizando o Arduino.

Silva (2017) traz um levantamento de pesquisas, envolvendo robótica educacional, publicadas entre os anos de 2012 à 2016 no SBIE (Simpósio Brasileiro de Informática e Educação) e na RBIE (Revista Brasileira de Informática na Educação). Na Figura 1.1

tem-se a apresentação de um gráfico com as porcentagens das categorias aplicadas no ensino de programação no Brasil.

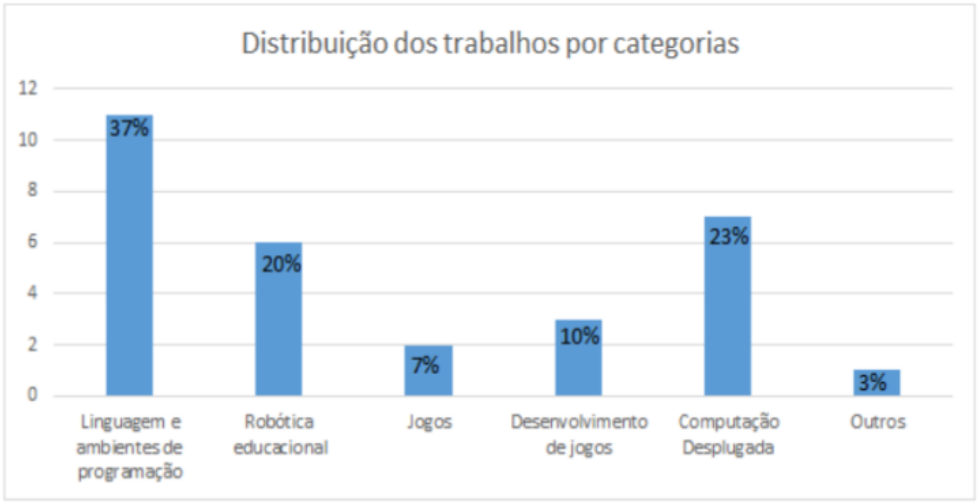

Figura 1.1 – Gráfico com porcentagens das categorias aplicadas no ensino de programação. Fonte: (SILVA, 2017)

Através da Figura 1.1 percebe-se a maior quantidade de trabalhos envolvendo linguagem e ambientes de programação. Ainda segundo Silva (2017), tal fato se dá pois os conteúdos colocados como mais importantes e, consequentemente, mais abordados nas aulas são: introdução a lógica de programação, conceitos de algoritmos e estruturas de linguagem de programação.

Dentre os trabalhos desenvolvidos no Brasil podemos citar primeiro Castro (2008), que propõe um software educacional para ensino da robótica denominado RoboEduc, criado para ser utilizado por crianças do ensino fundamental, podendo ser ensinados conceitos de robótica e programação. Já Siqueira e Vallim (2011) apresenta um módulo baseado em hardware reconfigurável, uma estrutura de módulos com motores, sensores e comunicação USB com o computador para a programação.

Outros trabalhos com propostas de desenvolvimento de kits são Silva e Scherer (2013), Kato, Braga e Pazmino (2015) e Busson (2017). Sendo eles um protótipo usando o Arduino, na sequencia um kit de um braço manipulador robótico e por último um kit didático mais completo, com diversos componentes e o uso do Arduino Mega.

No mercado atual é possível encontrar diversos produtos para diferentes aplicações dentro do ensino de programação e robótica. A seguir estão listados e detalhados alguns kits de robótica.

1.4.1 LEGO *Mindstorm*®

O kit LEGO *Mindstorm*, apresentado na Figura 1.2, é comercializado pela empresa dinamarquesa Lego, atualmente o kit está em sua terceira versão (EV3), que inclui componentes eletrônicos (microcontrolador com 4 portas de entrada e 4 portas de saída, sensor de toque e diversos módulos), estruturais (engrenagens, eixos vigas e polias) e a possibilidade de controle do dispositivo através de *smartphones* ou *tablets* por *Bluetooth*.

Figura 1.2 – Ilustração do Kit Lego Mindstorm EV3. Fonte: (LEGO, 2021)

A programação do dispositivo é gráfica no *software* da própria empresa, que consiste em arrastar e soltar ícones com o uso de um computador, como apresentado na Figura 1.3 a seguir.

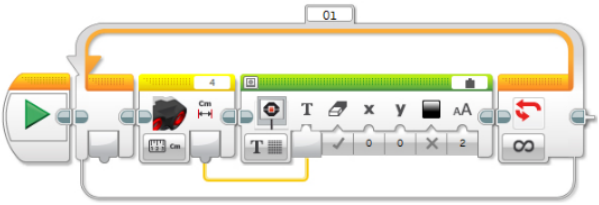

Figura 1.3 – Exemplo de programa construído para o Lego Mindstorm EV3. Fonte: (JUNIOR; GUEDES, 2015)

1.4.2 Modelix Robotics

O kit Modelix Robotics visa o desenvolvimento de projetos de robótica para diferentes níveis educacionais, envolvendo do ensino infantil até o ensino médio e superior. Em sua versão 3.6, o kit é composto por um microcontrolador, displayLCD, joystick, controle remoto, fonte de alimentação e diversos outros componentes, tais como sensores, atuadores e componentes estruturais.

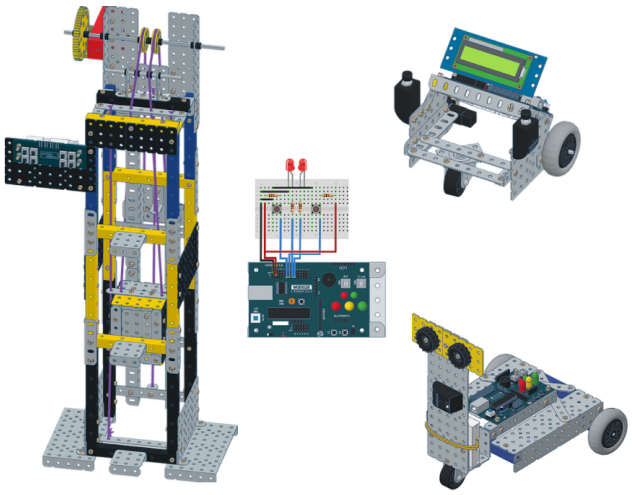

Figura 1.4 – Exemplo de projetos do Kit Modelix. Fonte: (MODELIX, 2021)

Para programação o kit conta com o *software Modelix System Pro*. Ele possui dois modos de utilização, o modo de programação baseado na construção de fluxogramas e um modo de simulação, ambos executados por meio de um computador.

1.4.3 OWIKIT

Os kits são da empresa *OWI Incorporated* que procurava uma maneira de trazer sua extensa experiência em tecnologia para a comunidade educacional de maneira acessível. Vale ressaltar que tais kits permitem a montagem de apenas um modelo em particular,normalmente sem condições de programação e, habitualmente, operado por um controle.

1.4.4 TI-Robotics-System-Learning-Kit (TI-RSLK)

Surgindo como uma alternativa a kits voltados para a robótica educacional e disponibilizado por umas das maiores fabricantes e distribuidoras de componentes eletrônicos do mundo a *Texas Instruments*.

O kit, que pode ser obersavado na Figura 1.5 é composto por diferentes módulos, como o chassi do robô, a placa-base de conexão dos diferentes periféricos, motores adaptados a rodinhas. E em uma versão completa o kit conta com módulos de comunicação *Wifi* e *Bluetooth*, sensores de distância, display LCD e um sistema de áudio.

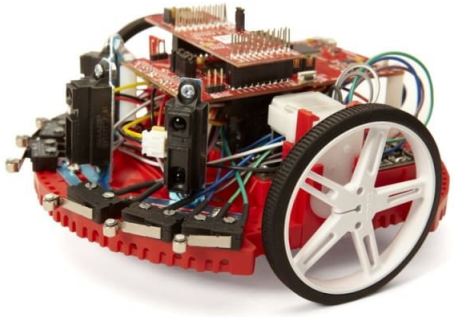

Figura 1.5 – Kit de desenvolvimento TI-RSLK®. Fonte: (BASSANI, 2020)

1.4.5 Makeblock

A *Makeblock* é uma recente empresa fundada em 2013 pelo chinês Jasen Wang. A empresa se dedica a produtos para o ensino com a metodologia STEAM, com uma variedade muito grande, indo desde impressoras 3D, cortadoras a laser até kits completos com diversos componentes eletrônicos. Dentre seus produtos podemos destacar três, são eles:

- ***Neuron***: kit com um conjunto de blocos eletrônicos, com encaixes por ímã, com diferentes tipos de sensores e atuadores. Este kit (Figura 1.6(a)) tem o diferencial da possibilidade de ser programado e controlado por um aplicativo mobile, além de ser compatível com peças de LEGO.

- ***mBot***: passando por diversos níveis de complexidade, a linha *mBot* conta com 3 modelos. Todas as versões do produto são baseadas em soluções que fazem uso de conjuntos de módulos e sensores programáveis capazes de proporcionar a experiência de construção de protótipos criativos. O modelo mais básico, o kit *mBot* (Figura 1.6(b)) promete um design atrativo e de fácil montagem. O conjunto conta com três modos nativos: desvio de obstáculos, seguidor de linha e controle manual, sendo este último aprimorado pela compatibilidade com controles remotos. Os softwares da *Makeblock* permitem desde a adequação ao uso do controle remoto, passando por programação em blocos, chegando até em códigos desenvolvidos em linguagem para a plataforma Arduino, pois seu microcontrolador é um chip *ATmega328*. Este kit básico acompanha sensor de luminosidade, ultrassônico e um seguidor de linha, além de um receptor IR. Já em termos de conectividade, o mBot tem suporte a conexões Bluetooth.

- ***HaloCode***:Este módulo é voltado para um público de idade um pouco mais avan-

Figura 1.6 – Produtos *Makeblock*: *Neuron*(a) e *mBot*(b). Fonte: Adaptado de (MAKEBLOCK, 2021)

çada. O *HaloCode* (Figura 1.7) é uma placa equipada com um ESP32 embutido e conectada a LEDs RGB programáveis, sensores de toque, sensor de movimento, permitindo a conexão *wireless* por meio de *Wifi* e *Bluetooth* para a programação do conjunto de infinitas maneiras desejadas, tudo isso com o auxílio de uma interface de programação em blocos.

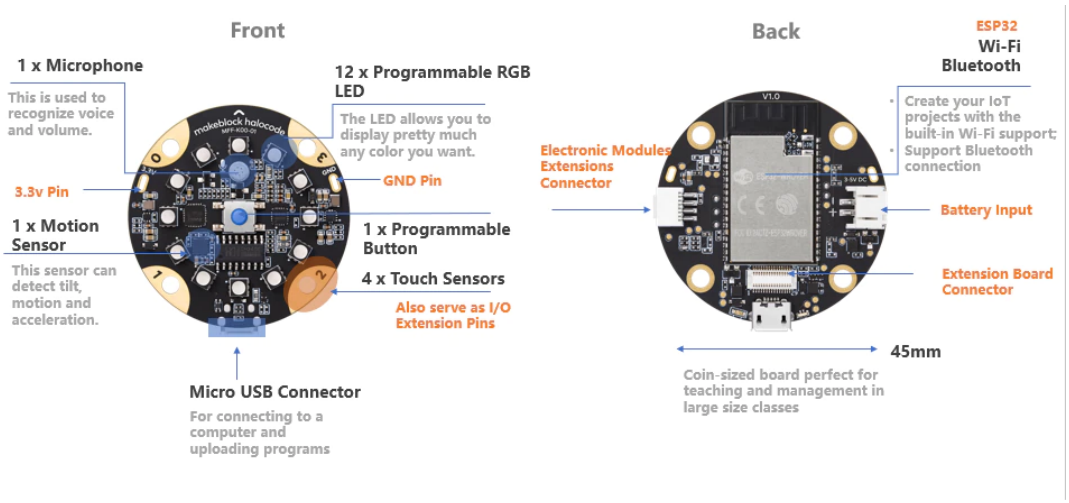

Figura 1.7 – Produto *HaloCode Makeblock*. Fonte: Adaptado de (MAKEBLOCK, 2021)

1.4.6 *Sphero BOLT*

Sphero BOLT é um robô de carcaça transparente no formato de uma esfera do tamanho aproximado de uma bola de beisebol, como demonstrado na Figura 1.8, adaptado a um pequeno painel de LEDs coloridos, completamente programável, acompanhado de uma série de periféricos, como sensores de luminosidade e infravermelho, giroscópio e acelerômetro. Para a programação do robô, o fabricante também proporciona softwares compatíveis, com foco principal em programação por meio de blocos (BASSANI, 2020).

Figura 1.8 – Modelo *Sphero BOLT Coding Robot*. Fonte: Adaptado de (SPHERO, 2021)

1.4.7 Microduino

A Microduino é uma multinacional recente de tecnologia educacional. Ela desenvolve kits e softwares para o ensino de programação, eletrônica e robótica. Pode-se citar 3 dos seus principais produtos:

- ***mCookie***: Com diversos sensores e atuadores, o mCookie foi desenvolvido para o ensino e aprendizado de robótica e programação de maneira fácil e intuitiva. Ele é compatível com diferentes linguagens, como o *Scratch*, que é feita em blocos, até mesmo C++ e *Python*. Dessa forma, a linha é ideal tanto para professores que já dominam a área de tecnologia, como para os que ainda estão iniciando. O kit pode ser programado pelo software *mDesigner* da própria empresa ou pela IDE do Arduino.

Figura 1.9 – Modelo *mCookie*. Fonte: Adaptado de (MICRODUINO, 2021)

- ***Itty Bitty Buggy***: Este kit conta com um microcontrolador, dois motores e alguns poucos sensores. Itty Bitty Buggy possui diversas peças campatíveis com peças LEGO para o desenvolvimento de diferentes projetos. No entanto, a fabricante destaca 5 projetos e modos de operação, como demonstrado na Figura 1.10. Um diferencial deste produto é a possibilidade de conexão *Bluetooth* com *smartphones* ou *tablets* para o controle ou programação, utilizando o aplicativo da própria empresa.

Figura 1.10 – Modelo *Itty Bitty Buggy*. Fonte: Adaptado de (MICRODUINO, 2021)

- ***IdeaBit***: Este produto é um proposta da *Microduino* para ser implementado em projetos *Maker*. O IdeaBit é uma placa programável através do software *mDesigner* da própria empresa, com 12 sensores e atuadores integrados, como demonstrado na Figura 1.11.

1.4.8 Análise comparativa entre os kits

Tendo observado os trabalhos e os diferentes kits de robótica educacional, fica evidente que existem muitas abordagens na literatura a respeito do ensino de programação e robótica. Dentre os kits disponíveis no mercado há algumas semelhanças, como por exemplo os tipos de sensores e atuadores integrados. Componentes como LEDs, botões, módulo emissor de som, motores, sensor *touch* e sensor de distância são os mais comuns

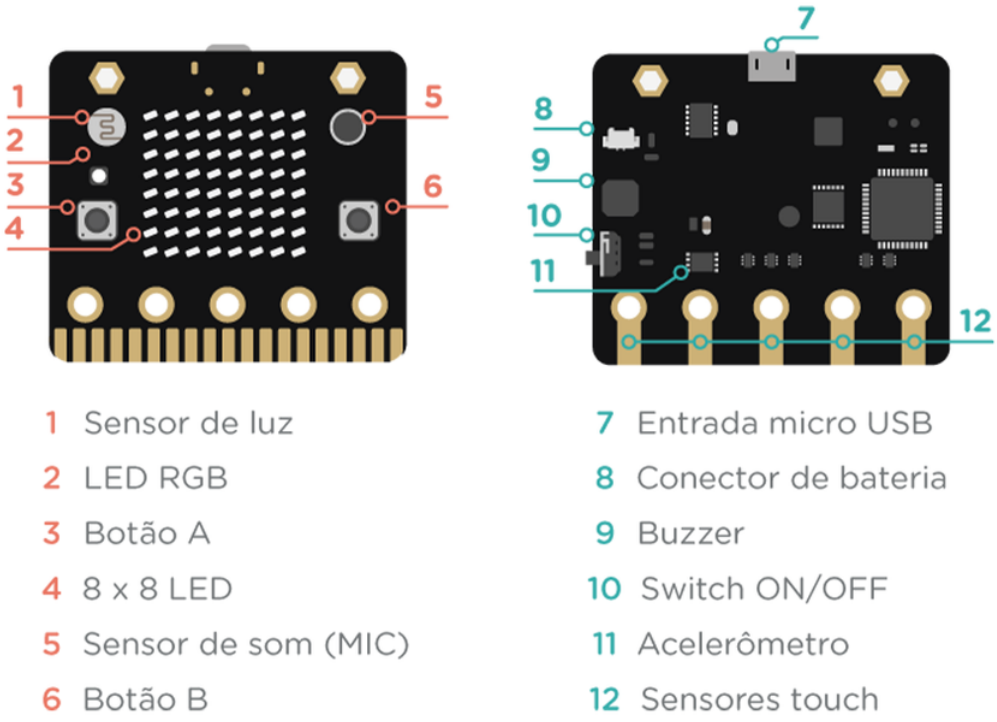

Figura 1.11 – Modelo *IdeaBit*. Fonte: Adaptado de (MICRODUINO, 2021)

entre os modelos, fato que contribuiu para a escolha dos componentes do módulo a ser desenvolvido neste trabalho.

Um fator interessante a ser analisado se trata do material pedagógico de apoio ao uso dos kits. Neste quesito, destacam-se os kits Lego *Mindstorm*, *Modelix Robotics* e os kits da *Microduino*, por possuírem documentação em português e um material para auxiliar os professores durante a aula.

Outro ponto de análise são os preços para aquisição dos kits, como demonstrado na Tabela 3.2. O custo elevado destes kits faz com que a aplicação deles nas escolas públicas se torne inviável. Além disso, diante dos valores, a compra e uso doméstico de algum kit e também a aplicação de práticas com um kit para cada aluno dentro das escolas particulares se torna infactível.

Somado a estes outros fatores, tem-se que poucos modelos no mercado possibilitam a programação do kit por meio de um *smartphone*. Portanto, diante destas informações é que este trabalho propõe o desenvolvimento de um módulo, que possibilite diferentes práticas de programação e robótica. Com o diferencial de entregar também um aplicativo com conteúdos sobre o ensino de programação e componentes de robótica, além da possibilidade de se programar o módulo através do celular.

Tabela 1.1 – Tabela de comparação dos valores dos kits.

Kit	Valor	Permite Programação por mobile
LEGO *Mindstorm*	R$3.994,99	Não
Modelix	R$ 1029,90	Não
OWIKIT - Kit STEM	R$964,39	Não
TI-RSLK	R$1037,37	Não
Makeblock mBot	R$615,19	Sim
Makeblock Neuron	R$801,13	Sim
Makeblock HaloCode	U$39,99[1][htb!]	Sim
Sphero	R$1232,44	Não
Microduino mCookie	R$2.100,00	Não
Microduino IdeaBit	R$199,00	Não
Microduino Itty Bitty Buggy	R$499,00	Sim
Ozobot Bit	R$599,00	Não
Ozobot Evo	R$979,00	Sim

1.5 Metodologia

A metodologia adotada na elaboração do trabalho foi pensada de forma que a evolução dos resultados ocorresse gradativamente e para isso, ela foi dividida em algumas partes.

Para a realização da pesquisa, inicialmente foi trabalhado a fase decisória, na qual o autor realizou uma pesquisa exploratória e descritiva sobre o que existe no mercado, envolvendo kits didáticos para ensino de programação, e como tem sido aplicado este ensino nas escolas do Brasil. Nesta etapa também foi avaliado a situação das escolas públicas em se tratando da aplicação de aulas de programação e recursos existentes.

Em paralelo a etapa mencionada anteriormente também foi realizado um levantamento e definição dos componentes utilizados para construção do módulo, tal como o seu layout e circuito eletrônico.

Dentro do desenvolvimento do aplicativo, foi objeto de pesquisa a aplicação da linguagem de programação *JavaScript* e o uso do *framework* React Native. Além disso, antes da fase construtiva, foi planejado o layout do aplicativo e o diagrama de Caso de Uso. Antes da fase construtiva, uma pesquisa também foi feita definindo o microcontrolador utilizado para o módulo.

[1] Produto ainda não disponível no Brasil.

Por fim, a última parte foi a mais longa do trabalho e consistiu na elaboração do dispositivo e do aplicativo. O início dessa etapa se deu com a conexão dos componentes ao microcontrolador seguida pela programação do mesmo para realização de testes. Após isso, projetou-se o circuito responsável pela alimentação do dispositivo e finalizou com a definição da posição de cada componente no layout do módulo. Em paralelo a esta última etapa se deu a criação do aplicativo.

1.6 Organização do Documento

Este documento está dividido em 5 capítulos. Esse capítulo, aborda os conceitos introdutórios responsáveis pela escolha do tema a ser estudado. E definido o problema, em seguida, são demonstrados os objetivos do trabalho, a motivação para realizá-lo, o estado da arte e a metodologia utilizada durante o desenvolvimento do projeto.

No capítulo seguinte, são apresentados os fundamentos que embasam o trabalho, sendo apresentado a revisão bibliográfica e a fundamentação teórica.

Em seguida, no terceiro capítulo, o desenvolvimento e os resultados são apresentados. Nessa parte, são apresentadas as etapas executadas para realização do projeto de maneira esquemática para facilitar a visualização da estratégia de desenvolvimento adotada.

Por fim, o último capítulo apresenta as considerações finais do trabalho em conjunto com as propostas de continuidade.

Capítulo 2

Revisão de Literatura

Este capítulo apresenta uma revisão bibliográfica sobre os temas que permeiam este trabalho, demonstrando a evolução tecnológica dos componentes utilizados e também o estado da arte. Por fim, conceitos para o desenvolvimento do mesmo são apresentados.

2.1 Revisão Bibliográfica

A partir da década de 1940 temos os primeiros computadores digitais, que possuíam certas limitações, como por exemplo, a presença de apenas dois níveis, no qual um era feita a programação e no outro nível, chamado de nível lógico digital, era executado os programas, sendo este último com circuitos complicados e não confiáveis. Em 1951, Maurice Wilkes trouxe a possibilidade de um computador ter 3 níveis, contando agora com um microprograma, cuja função fosse executar programas por interpretação (TANENBAUM, 2007).

Com o passar dos anos foi-se somando muitos avanços e descobertas de diferentes tipos de computadores. Segundo Tanenbaum (2007), podemos dividir em diferentes gerações de computadores, com base nas tecnologias utilizadas, para entendermos melhor o histórico e como chegamos ao ponto que estamos hoje, estas gerações são:

- Geração zero: computadores Mecânicos (1642 - 1945);
- Primeira geração: Válvulas (1945 - 1955);
- Segunda geração: transistores (1955 - 1965);
- Terceira geração: circuitos integrados (1965 - 1980);

- Quarta geração: integração em alta escala de transistores em um único chip (1980);
- Quinta geração: computadores baseados em inteligência artificial;

Além disso, com o avanço da microeletrônica e com a possibilidade de miniaturizar e integrar diversos circuitos em um único chip surgem os primeiros microcontroladores. O primeiro deles começou a ser projetado em 1969, com a necessidade de uma empresa japonesa, a BUSICOM, no seu projeto de uma calculadora eletrônica. Os engenheiros dessa empresa embarcaram para os Estados Unidos com o objetivo de encontrar Marcian Hoff, da *Intel Corporation*, para expor as necessidades do dispositivo. Tempos depois, em 1971, após comprar a licença da empresa japonesa a Intel lançou o 4004, um controlador de 4 bits que conseguia processar 6000 operações por segundo (6kHz) (AYCOCK, 2017).

Com o avanço desta tecnologia, surge empresas concorrentes, como Motorola e MOS Technology, lançando modelos diferentes. Além disso, durante os anos 9, surge a possibilidade de utilização de memórias eletronicamente apagáveis, ou seja, os microcontroladores agora passam a possuir memórias ROM (EEPROM) e, assim, poderiam ser programados, apagados e reprogramados (AYCOCK, 2017). Com este avanço os dispositivos agora poderiam receber atualizações de softwares sem a necessidade da troca do microcontrolador.

Os microcontroladores têm se tornado cada vez mais potentes e menores. Sendo utilizados em diferentes áreas, como: automotiva, iluminação, comunicação, dispositivos de baixo consumo de energia, eletrodomésticos e uma infinidade de outros aparelhos. De acordo com Lima e Villaça (2012) um microcontrolador é "um sistema microprocessado com várias funcionalidades (periféricos) disponíveis em um único chip". Tal fato que o difere de um microprocessador, que não possui memórias de programa, de dados e RAM, temporizadores e circuitos de clock embutidos.

Atualmente, um forte exemplo da utilização das tecnologias de um microcontrolador é o Arduino. O Arduino, que originalmente foi criado como uma ferramenta para projetos de computação para designers e estudantes, é uma plataforma de prototipagem eletrônica de hardware livre e de placa única, projetada com um microcontrolador (EVANS; NOBLE; HOCHENBAUM, 2013).

O Arduino surgiu no *Interaction Design Institute*, na Itália em 2005. Na circunstância o professor Massimo Banzi buscava uma alternativa barata e de fácil utilização para seus alunos trabalharem com tecnologia.E em discussão com outros professores, pesquisadores e o auxílio de um engenheiro, foi criado o Arduino. Diante da facilidade de utilização e

seu baixo custo a placa em questão logo se popularizou e, atualmente, alcança centenas de milhares de unidades vendidas e várias versões de placas, como representado na Figura 2.1. Sendo o mais popular o Arduino Uno, que conta com um microcontrolador ATmega328P.

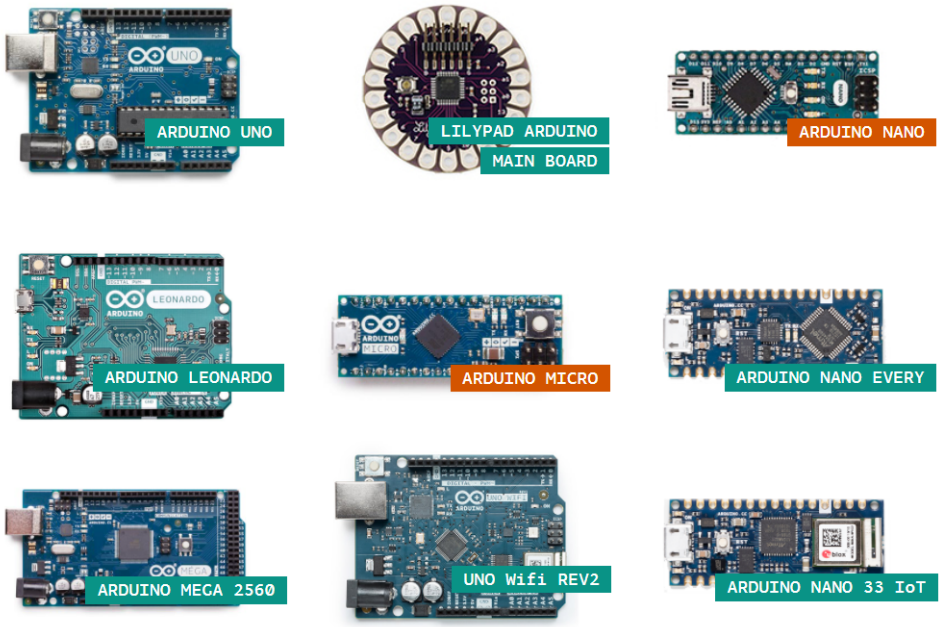

Figura 2.1 – Modelos de Arduino. Fonte (ARDUINO, 2021)

Devido a popularização do Arduino várias outras placas com microcontroladores surgiram. Como exemplo, temos o ESP32 da *Espressif Systems*, mostrado na Figura 2.2. Segundo Kurniawan (2019), "esse dispositivo é um chip de baixo custo que consiste em um microcontrolador com protocolos de comunicação *Wi-Fi* e *Bluetooh*, o que torna possível, a utilização em conjunto com um aplicativo de Internet das Coisas (IoT)."[1].

Os microcontroladores trouxeram consigo muitas possibilidades e facilidades, não somente no meio industrial. De acordo com os autores Tocci, Widmer e Moss (2011):

> "Os estudantes de hoje têm de ser expostos a estas ferramentas modernas, mesmo em um curso introdutório. É responsabilidade de todo educador encontrar a melhor maneira de preparar os estudantes para o que eles encontrarão na vida profissional."

Este trabalho, portanto, se propõe a fazer uma investigação e pesquisa para identificar o melhor modelo de microcontrolador, dentro os diversos modelos, para ser implementado em um módulo didático para o ensino e a prática de programação e robótica, não somente dentro das escolas mas também nas casas dos alunos. E, segundo (BOLTON, 2010), ao

[1] Do inglês Internet of Things.

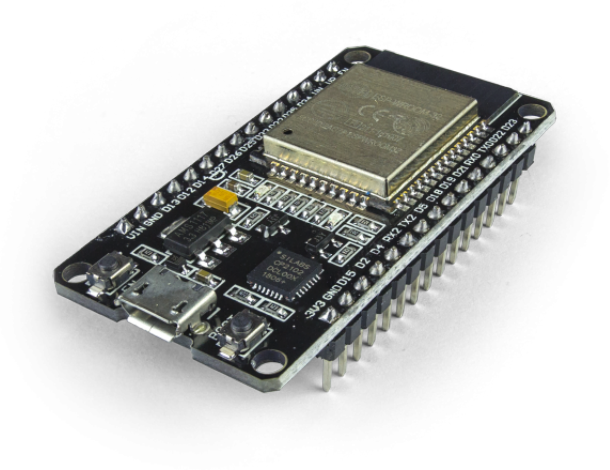

Figura 2.2 – Placa ESP WROOM 32. Fonte (ROBOCORE, 2020)

selecionar o microcontrolador para desenvolvimento de um projeto deve-se considerar os seguintes fatores:

- Número de pinos de entrada/saída;
- Interfaces necessárias, como por exemplo o número de saídas PWM;
- Requisitos de memória;
- Número de interrupções necessárias;
- Requisitos de velocidade de processamento

Outra tecnologia que mostrou sua evolução ao longo dos anos foi a comunicação sem fio entre dispositivos. Dentre os vários tipos existentes hoje, o precursor foi a comunicação via radiofrequência. Tudo começou em 1895, quando Guglielmo Marconi conseguiu transmitir sinais em código Morse no jardim de sua casa, sem o uso de fios. Em 1904, o padre brasileiro Landell de Moura registrou patentes para um telefone sem fio e um telégrafo sem fio, equipamentos precursores do rádio. Finalmente, em 1920, a exploração regular do serviço de radiodifusão tem início nos Estados Unidos, e o rádio não-comercial se estabeleceu na Inglaterra dois anos depois. A partir de 1927, começa a era eletrônica do rádio, eliminando a necessidade de captação do áudio pelo microfone do toca-discos.

As primeiras emissoras em frequência modulada, com qualidade melhor que as de amplitude modulada, começaram a fazer suas transmissões publicamente a partir de 1976 (TAPARELLI, 2002).

Alguns anos depois, em 1995, surgiu a primeira tecnologia *Wireless* capaz de conectar telefones móveis e outros aparelho por meio de ondas de rádio, o *Bluetooth*. Segundo Lugli e Sobrinho (2012), esse protocolo é capaz de transmitir sinais de dados e voz e "se desenvolveu com uma frequência de rádio aberta, operando na faixa ISM (*Industrial, Scientific and Medical*), com comunicação por salto de frequência FH-CDMA (*Frequency Hopping - CodeDivision Multiple Access*), permitindo proteção e fazendo com que a frequência seja dividida em vários canais."

Dois anos depois, em 1997, com a criação do padrão IEEE 802.11, surgiu o *Wi-Fi*. Este padrão define as normas para o uso da rede sem fio e segundo Stefanuto, S. e Torres (2016), "sua transmissão é realizada por radiofrequência, espalhando o sinal pelo ar podendo chegar a centenas de metros, dependendo do equipamento e dos obstáculos, tais como árvores e paredes."

2.2 Fundamentação Teórica

Neste capítulo será apresentado os fundamentos teóricos das principais ferramentas e elementos que envolvem o desenvolvimento deste trabalho.

2.2.1 Arduino Uno

O Arduino é uma plataforma que oferece diversas vantagens. Dentre elas pode-se citar: baixo custo, ambiente de programação simples e claro, software de código aberto e extensível e um hardware extensível e de código aberto. Como o próprio fabricante descreve:

> Arduino é uma plataforma eletrônica de código aberto baseada em hardware e software fáceis de usar. As placas Arduino são capazes de ler entradas - luz em um sensor, um dedo em um botão ou uma mensagem do Twitter - e transformá-la em uma saída - ativando um motor, ligando um LED, publicando algo online. Você pode dizer à sua placa o que fazer enviando um conjunto de instruções para o microcontrolador da placa. Para fazer isso, você usa a linguagem de programação Arduino (baseada em Wiring), e o Arduino Software (IDE) , baseado em Processing. (ARDUINO, 2021)

Dentre as placas Arduino a mais utilizada é o Arduino Uno (Figura 2.3). Esta placa possui o microcontrolador ATmega328P, 14 pinos de entrada e saída digital, 6 entradas analógicas, conexão USB e um conector de alimentação.

Figura 2.3 – Arduino Uno. Fonte (ARDUINO, 2021)

2.2.2 Arduino Nano

O Arduino Nano é uma versão bem semelhante ao Arduino Uno. Os diferenciais desta versão são tamanho reduzido e compacto, mais barato, 2 portas analógicas a mais, em relação ao Uno, e um conector USB Mini-B (ARDUINO, 2021). Para projetos em que é necessário uma placa de tamanho reduzido o Arduino Nano (Figura 2.4) é uma excelente opção.

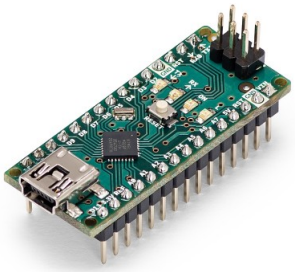

Figura 2.4 – Arduino NANO. Fonte (ARDUINO, 2021)

2.2.3 ESP

Os microcontroladores da família ESP tem ganhado popularidade ao longo dos anos, principalmente para aplicações que exigem uma comunicação sem fio. Como dito na seção anterior, um modelo mais atual e bastante utilizado em aplicações de IoT é o ESP32. Essa

placa conta com um processador *dual-core*, cerca de 500 kBytes de memória SRAM, *Wi-Fi* e *Bluetooh 4.2* de forma integrada além de possuir 25 sinais de GPIO, sinais PWM e conversores AD, como mostrado na Figura 2.5. Além disso, tem-se os diferenciais como o sensor de temperatura para monitoramento da temperatura da placa e a interface para sensor de toque, facilitando o uso de teclados e botões capacitivos (ESPRESSIF, 2021).

Para realizar a programação do dispositivo é possível utilizar várias IDEs, como a do CodeBlocks, Netbeans, Lua e principalmente a do Arduíno.

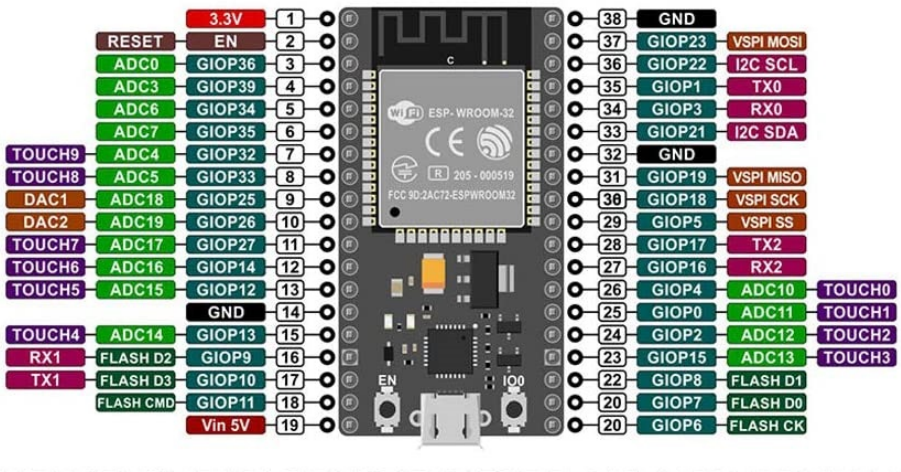

Figura 2.5 – Pinagem ESP32. Fonte: (COELHO, 2021)

Este trabalho também conta com o estudo e possíveis testes utilizando outros modelos ESP. Um deles será o ESP8266, por ser uma versão anterior do ESP32 possui menos funcionalidades mais ainda se faz uma boa opção devido ao seu baixo custo. Dentre as características do ESP8266 temos wifi embutido no chip, tensão de operação de 3,3V, 32 KBytes de RAM para instruções, 96KBytes de RAM para dados e 64KBytes de ROM para boot (ESPRESSIF, 2021).

Será estudado a aplicação do módulo ESP8266 NodeMcu ESP-12E, demonstrado na Figura 2.6, e também o módulo ESP-01 em conjunto com um Arduino.

2.2.4 React Native

O *React Native* é um *framework open source* baseado no React, desenvolvido pela equipe do Facebook, que possibilita o desenvolvimento de aplicações mobile, tanto para Android, como para iOS, utilizando *Javascript* (REACTNATIVE, 2021).

O *React Native* foi desenvolvido pela equipe do Facebook. Conforme esclarece Araujo (2019), a estrutura foi lançada no ano de 2015, com a missão de facilitar o desenvolvimento

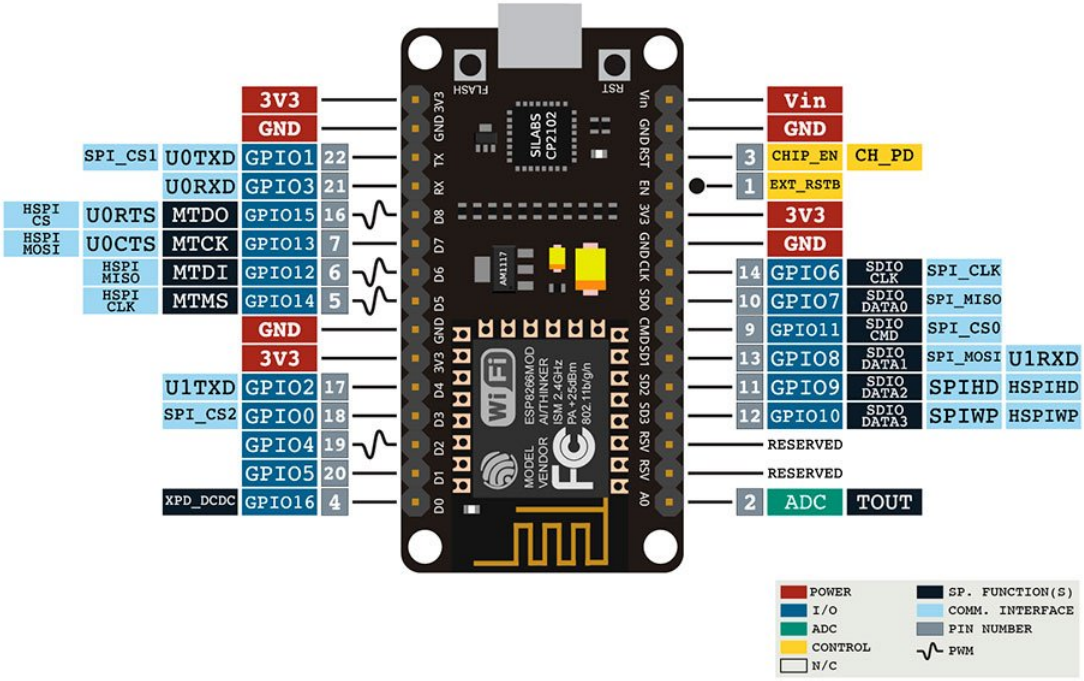

Figura 2.6 – Pinagem ESP8266 NodeMcu ESP-12E. Fonte: (ESPRESSIF, 2021)

de interfaces Web complexas. Porém, a empresa já trabalhava no React há alguns anos, mesmo antes de abrir o código para a comunidade. Eles buscavam resolver um dos maiores problemas da Web, que é o custo operacional de manipular o Document Object Model (DOM), uma estrutura em formato de árvore que descreve a organização e o estado atual dos elementos de um documento (ARAUJO, 2019).

Ainda segundo Araujo (2019), no React os componentes *Javascript* são declarados, utilizando uma série de primitivas embutidas que são suportadas tanto pelo sistema iOS como pelo Android. Conforme destaca (BECKER, 2021), essa característica não faz com que a ferramenta seja surpreendente nem inovadora, pois existem inúmeros *frameworks* que empacotam aplicações web dessa maneira. Porém, o autor chama a atenção para o fato do *React Native* apresentar uma diferença interessante: nele, todo o código desenvolvido é convertido para a linguagem nativa do sistema operacional.

O autor Becker (2021) ainda elenca várias vantagens que uma aplicação nativa apresenta sobre uma aplicação web mobile: experiência do usuário fluída; carregamentos em geral mais rápidos; melhor integração entre funções do celular como câmera, giroscópio, entre outras; maior segurança; melhor performance em geral.

Informações retiradas do site oficial da plataforma demonstra que já em 2018, o *React Native* tinha o segundo maior número de colaboradores para qualquer repositório no GitHub. Atualmente, ele é apoiado por contribuições de indivíduos e empresas em

todo o mundo, incluindo Callstack , Expo , Infinite Red , Microsoft e Software Mansion (REACTNATIVE, 2021). Enfim, observa-se que o *React Native* é um *framework* bastante aceito pelos profissionais que trabalham na área de desenvolvimento de software, principalmente no setor mobile.

2.2.5 Java Script

A linguagem base utilizada para desenvolvimento dos aplicativo com o *React Native* é o *Java Script*. A linguagem de programa *JavaScript* foi desenvolvida pelo *Netscape* em 1995. Trata-se de uma linguagem de programação leve, interpretada e baseada em objetos com funções de primeira classe. Não se deve confundir esta linguagem com *Java*, ambas marcas são registradas da *Oracle* nos Estados Unidos da América e em outros países. No entanto, as duas linguagens de programação possuem sintaxe, semânticas e usos muito diferentes (MDN, 2021).

Assim sendo, de acordo com Stellzer (2014), esta linguagem tem a capacidade de tornar os sistemas mais interativos. O autor descreve algumas das principais características do *JavaScript*, tais como: as variáveis oferecerem tipagem dinâmica; ser uma linguagem interpretada e, assim sendo, seu código-fonte não precisa ser compilado.

2.2.6 Visual Studio Code

O Microsoft Visual Studio Code é uma ferramenta para edição de código fonte desenvolvido da Microsoft que, atualmente, está com uma presença muito marcante no cenário de desenvolvimento de aplicações. Dentre os motivos para tal fato, tem-se a licença para esse editor de textos ser gratuita e seu código fonte ser disponibilizado no GitHub.

O Visual Studio Code possui suporte à ASP.NET, Node.js, PHP, Ruby, Python, entre outras. E a ferramenta não se trata de uma cópia do Visual Studio Enterprise, mas, sim, um editor multiplataforma e rápido, semelhante a outros como o Sublime Text, Atom ou Brackets (FOGAÇA, 2018). O desenvolvimento do aplicativo deste trabalho será realizado utilizando o VS Code como a principal ferramenta para elaboração dos códigos.

Capítulo 3

Desenvolvimento e Resultados

Este capítulo tem por finalidade a descrição das principais etapas elaboradas durante este trabalho, visando alcançar os objetivos propostos. Objetivando uma maior organização textual, realizou-se uma divisão por tópicos. A primeira seção descreve os materiais e tecnologias utilizados no desenvolvimento do projeto, detalhando os procedimentos e parâmetros adotados.

3.1 Materiais e métodos

Para este trabalho de conclusão de curso propõe-se a utilização de diferentes materiais para projetar, simular, construir e testar o dispositivo e o aplicativo. Nos tópicos a seguir estão representados os materiais e métodos escolhidos.

3.1.1 Eletrônica

Com base no estudo dos kits comercializados por diferentes empresas e também nos projetos apresentados no capítulo anterior, foi possível identificar os componentes que são comuns entre os produtos, tal como as possibilidades de práticas que cada kit permite.

A partir disso, foi selecionado alguns componentes para integrar o módulo desenvolvido neste trabalho, portanto, temos: 4 LEDs (vermelho, amarelo, azul e verde); um LED RGB; 2 *push buttons*; um potenciômetro de 1KΩ; um sensor LDR; um sensor ultrassônico (HC-SR04); um buzzer; um sensor de temperatura (DHT11); 3 botões *touch* e um módulo TM1637 Display de 7 segmentos de 4 dígitos. Além disso, o dispositivo poderá ser utilizado

para acionamento de um servo motor. Os componentes estão demonstrados na Figura 3.1 e possibilitarão a prática de diversas atividades e projetos diferentes.

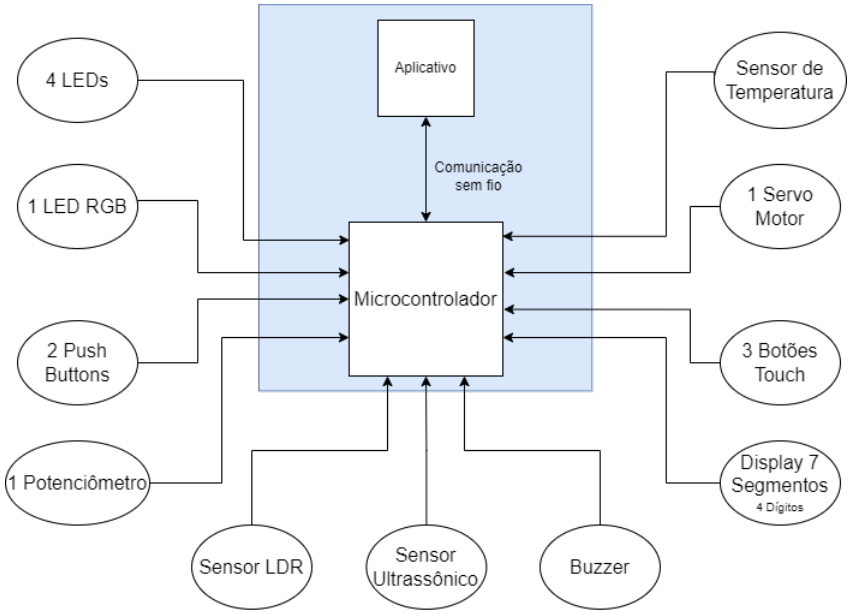

Figura 3.1 – Diagrama de descrição dos componentes integrados ao módulo. Fonte: Elaborado pelo autor.

3.1.1.1 Microcontrolador

Como um dos principais passos dentre as escolhas dos materiais tem-se a definição do microcontrolador a ser utilizado. Tal escolha foi realizada dentre as seguintes opções:

- Opção 1: Arduino Uno + Módulo Bluetooth HC-05;
- Opção 2: Arduino Nano + Módulo Bluetooth HC-05;
- Opção 3: Arduino Uno + ESP-01;
- Opção 4: Arduino Nano + + ESP-01;
- Opção 5: ESP8266 NodeMcu ESP-12E;
- Opção 6: ESP-WROOM-32.

Dentre os critérios que foram utilizados para a definição do microcontrolador teve-se o tamanho do hardware, o custo, número de portas para conexão de todos os componentes e também as características de comunicação sem fio.

Tabela 3.1 – Comparativo entre microcontroladores.

	Arduino Uno	Arduino Nano	ESP8266	ESP32
Processadores	1 núcleo	1 núcleo	1 núcleo	2 núcleos
Pinos I/O	23	22	17	38
Tensão de operação	5V	5V	3 a 3,6V	3 a 3,6V
V_{IN}	7 a 12V	7 a 12V	5 a 9V	5 a 9V
Corrente de consumo	15mA	19mA	80mA	80mA
WiFi	-	-	2,4 GHz	2,4 GHz
Bluetooth	-	-	-	BLE v4.2
Dimensão (mm)	68,58 x 53,34	18,5 x 43,2	25,5 x 49	26 x 50

Além das informações apresentadas na Tabela 3.1 outro importante critério avaliado foi o quanto o microcontrolador iria afetar financeiramente a construção do módulo, portanto foi utilizado as informações da Tabela 3.2 a seguir.

Tabela 3.2 – Comparativo de valores entre as opções de microcontroladores.

Opções	Valor[1]
Arduino Uno + Módulo Bluetooth HC-05	$R\$77,50$
Arduino Nano + Módulo Bluetooth HC-05	$R\$70,95$
Arduino Uno + ESP-01	$R\$66,78$
Arduino Nano + + ESP-01	$R\$60,23$
ESP8266 NodeMcu ESP-12E	$R\$31,35$
ESP-WROOM-32	$R\$49,49$

Foi realizado testes e estudos quanto ao processo de regravação dos microcontroladores, para verificar as possibilidades deste processo ser realizado através da comunicação sem fio com o aplicativo. Feito isto, foi possível validar a escolha do microcontrolador. Mesmo o ESP8266 tendo menor valor, devido as características de número de pinos e a possibilidade de conexão *Bluetooth* o ESP-WROOM-32 se mostrou a melhor opção, e o mesmo foi escolhido.

3.1.1.2 Alimentação

Como a proposta deste projeto envolve o uso do kit sem a utilização de um computador, se faz necessário um sistema de alimentação com capacidade de recarregamento.

[1] Os valores foram coletados do site: *www.wjcomponentes.com.br*

Para a definição da alimentação do dispositivo foi baseado primeiramente na carga suficiente para oferecer uma boa autonomia ao dispositivo. Com isso, após a definição do microcontrolador a ser utilizado, foi calculado a corrente consumida por cada componente do módulo e, a partir disso, estimado a corrente total necessária.

Tabela 3.3 – Tabela de corrente por componente.

Componente	Corrente
ESP 32	80mA
4 LEDs	20mA
LED RGB	20mA
Sensor de Temperaura DHT11	2,5mA
Display 7 Seg. TM1637	80mA
Sensor Ultrassom HC-SR04	15mA
Servo Motor	220mA
Buzzer	25mA
Módulo TP4056 (Circuito de Alimentação)	Desprezível
Módulo Conversor Boost (Circuito de Alimentação)	Desprezível
Circuito indicador de carga da bateria	Desprezível
Total	462,5mA

Portanto, a bateria adequada para as necessidades do projeto em questão deverá ser uma bateria recarregável, que permite o uso intenso do dispositivo durante mais de duas horas, possibilitando a utilização durante uma hora aula com tranquilidade.

Com base na Tabela 3.3 uma bateria adequada as necessidades do projeto é a *Li-ion* 18650 recarregável de 4,2V, mostrada na Figura 3.2. Tal decisão também foi tomada com base na facilidade de se encontrar tal modelo de bateria no mercado e no seu baixo valor. Esta bateria possui diversos modelos com diferentes cargas, porém, mesmo se utilizarmos o modelo de 2600mA ela irá fornecer uma autonomia de mais de 5h de uso intenso ao dispositivo. É importante pontuar que somente com os testes aplicando o circuito de alimentação no o módulo finalizado que esta informação poderá ser validada.

3.1.1.3 Módulo de carga

A fim de facilitar o recarregamento da bateria, sem a necessidade da retirada da bateria do dispositivo e o desmonte do mesmo, percebeu-se a necessidade de um módulo de carga. Portanto, o módulo definido foi o modelo TP4056, apresentado na Figura 3.3,

Figura 3.2 – Bateria recarregável 18650.

com entrada micro USB, pois este modelo, além de ter um baixo custo, conta com proteção de sobrecarga e término do ciclo de carga automático.

Vale ressaltar que tal módulo permite o carregamento da bateria utilizando um carregador de celular, porém o mesmo deve fornecer uma tensão de 5V e uma corrente mínima de 1A.

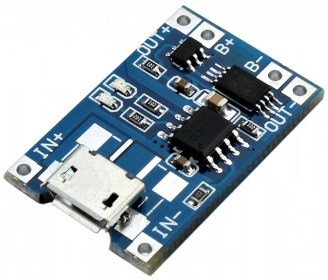

Figura 3.3 – Módulo Carregador de Baterias TP4056. Fonte: (COELHO, 2021)

3.1.1.4 Conversor Boost Step Up

O microcontrolador ESP32 possui três possibilidades de alimentação, uma pela entrada micro USB, outra pelo pino de 3,3V e pelo pino de 5V com uma tensão de 5v à 9v. Além disso, o módulo conta com componentes como o sensor ultrassom e servo motor que necessitam de uma alimentação acima de 5v, fato que a bateria *Li-ion* 18650 escolhida não consegue atender.

Para solucionar tal questão, foi adicionado ao final do circuito de alimentação um módulo conversor DC Boost step up ajustável, mostrado na Figura 3.4. Com isso todo o módulo, com o ESP32 e os componentes podem ser alimentados com 5V por este conversor e com 3,3v pelo próprio microcontrolador. Além disso, com o uso do conversor é garantido

que o módulo sempre será alimentado com 5v, mesmo quando a bateria estiver com sua capacidade e carga média.

Figura 3.4 – Conversor DC boost step up ajustável.

3.1.1.5 Outros

Além dos componentes citados nos tópicos anteriores, tem-se também a utilização de itens básicos de eletrônica, como:

- Resistores (220Ω, 470Ω, 10KΩ, 4,7KΩ, 8KΩ, 100kΩ, 48kΩ e 220kΩ);
- Chave On/Off;
- Transistores BC548;
- Porta bateria 18650;
- Placa de fenolite.

3.1.2 Estrutura

Para a construção da case do dispositivo optou-se por construir utilizando a impressão 3D. O material escolhido foi o filamento PLA, pois é um material de fácil impressão e não é prejudicial a saúde e ao meio ambiente, pois é obtido através de ingredientes naturais e recursos renováveis. Além disso, oferece ótimo custo-benefício.

3.1.3 Softwares

Para o desenvolvimento do trabalho, alguns *softwares* tem que ser utilizados. Sendo eles:

3.1.3.1 Arduino IDE

Para a programação do microcontrolador nos testes a IDE do Arduino será utilizada. Essa escolha se deve ao fato de o *software* ser gratuito e de código aberto. A linguagem de programação utilizada nessa IDE é o C++.

3.1.3.2 Visual Code Studio

Para a programação do aplicativo tem-se o *VS Code*. Ele é um *software* livre e de código aberto lançado em 2015.

3.1.3.3 React Native

Em conjunto com o *software* citado acima, o *framework React Native* também será utilizado na elaboração da aplicação. Como citado no capítulo anterior, essa ferramenta possibilita o desenvolvimento multiplataforma e facilita a construção de uma interface amigável ao aplicativo projetado.

3.1.3.4 SolidWorks

Esse programa de computação gráfica permite a criação de objetos com modelagem 3D a partir da tecnologia CAD 3D. Seu funcionamento se da a partir de bases de objetos sólidos, que podem ser modeladas para a criação de inúmeros arquivos diferentes. A escolha desse *software* se deu pela experiência do autor obtida ao decorrer do curso. Vale ressaltar que foi utilizado a versão de teste gratuito para este trabalho.

3.1.3.5 MultiSIM

Esse *software* foi utilizado na fase de desenvolvimento e simulação do circuito indicador de carga da bateria. Ele foi escolhido pelo fato de ser gratuito, de fácil utilização e por apresentar os componentes, instrumentos de medição e gráficos de forma simples.

3.1.3.6 PROTEUS

Proteus Design Suite é um software para criação de projetos eletrônicos, composto por uma suíte de ferramentas, incluindo captura esquemática, simulação e módulos de projetos de placas de circuito impresso, usadas principalmente para o projeto de circuitos integrados. Este *software* foi utilizado no projeto para criação do circuito e da placa do circuito indicador de carga e a placa do módulo.

3.2 O Módulo

A construção do módulo envolveu etapas como estudo do microcontrolador e componentes, planejamento do layout e posicionamento dos componentes, projeto elétrico e construção do circuito indicador de carga da bateria, testes e simulações, projeto elétrico e construção da placa do módulo e projeto da case do módulo. Após a fase de estudos, foi desenvolvido um esboço para verificar o layout que o módulo deveria ser produzido, a fim de facilitar seu dimensionamento e produção. O módulo foi projetado para ter como dimensões 8cm x 13cm. Abaixo segue a Figura 3.5 ilustrando o esboço.

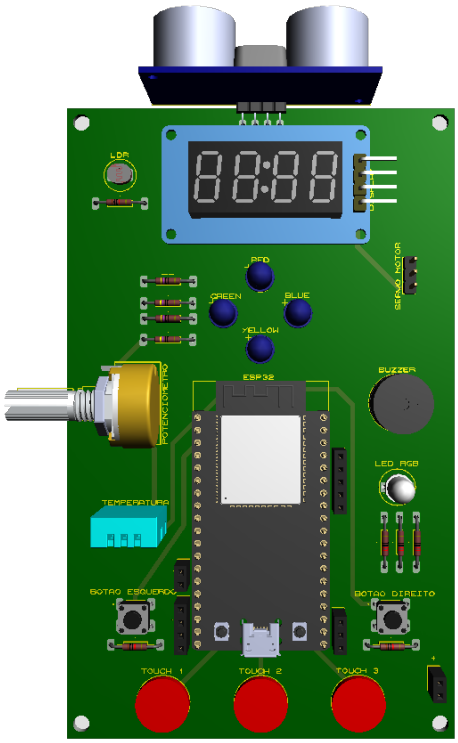

Figura 3.5 – Esboço do módulo.

3.2.1 Circuito Indicador de Carga

Objetivando oferecer maior segurança e facilidade para o usuário recarregar o módulo, buscou-se projetar um circuito que indicasse quando a bateria estiver com carga baixa, acendendo um LED vermelho como indicador.

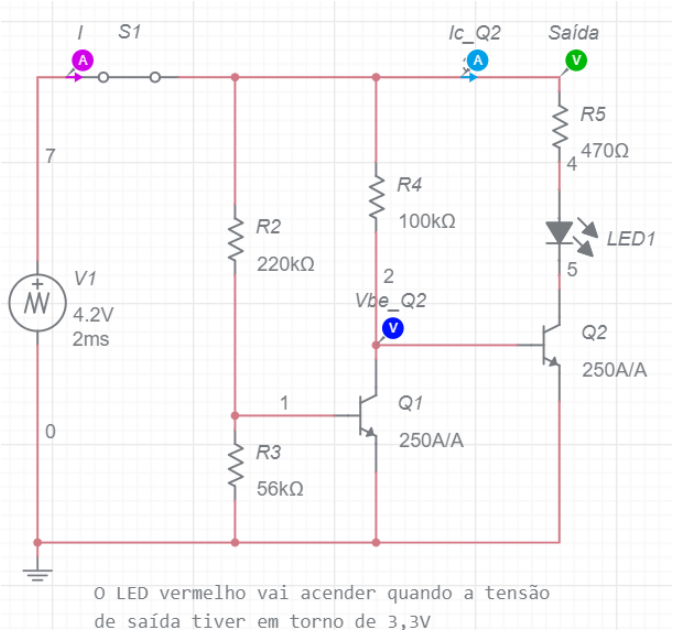

Figura 3.6 – Circuito de alimentação do dispositivo.

O circuito em questão, mostrado na Figura 3.6, contém 4 resistores (220KΩ, 48KΩ, 8KΩ, 100KΩ e 470Ω), 2 transistores BC548, uma chave *on-off* e 1 LED.

Para o projeto do circuito e simulações foi utilizado o *software MultiSIM* e na Figura 3.7 é mostrado curvas das tensões e correntes medidas.

No circuito o transistor utilizado está operando como uma chave. Com base na Figura 3.7 tem-se de verde a tensão que está sendo aplicada ao circuito e está saindo para o módulo conversor boost step up, de azul tem-se a tensão base-emissor aplicada no transistor conectado ao LED vermelho e as linhas pontilhadas são as correntes, sendo a azul claro a corrente aplicada no LED (coletor do transistor).

Portanto, o funcionamento do circuito consiste em quando a tensão da bateria for maior ou igual que 3,3V o transistor entra em modo de corte, pois sua tensão base-emissor fica menor que 0,5V e assim a corrente de coletor zera e o LED vermelho se apaga. Caso contrário, o LED ficará acesso, indicando uma carga baixa na bateria.

Após a etapa de projeção e simulação do circuito, partiu-se para a construção da placa. Para tal etapa foi utilizado o *software* PROTEUS e o projeto da placa pode ser

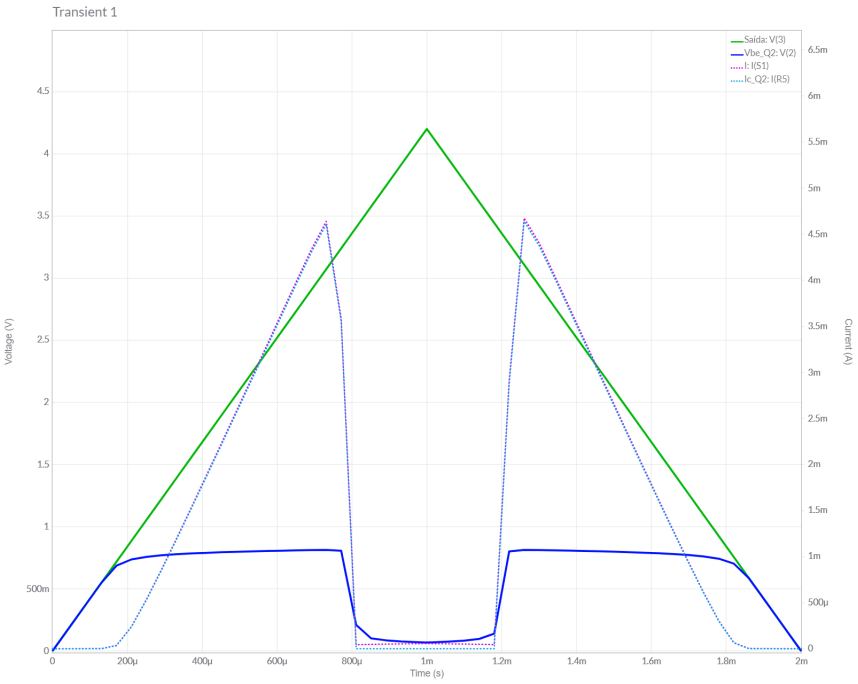

Figura 3.7 – Gráfico de simulação do circuito de alimentação do dispositivo.

visto no Apêndice A. A placa foi construída utilizando uma placa de fenolite e o método de transferência a frio, que consiste em usar acetona para transferir o circuito impresso em papel couche 180g para a chapa com cobre. Após este processo foi realizado a corrosão, furação e a soldagem dos componentes. O resultado final pode ser visto na Figura 3.8.

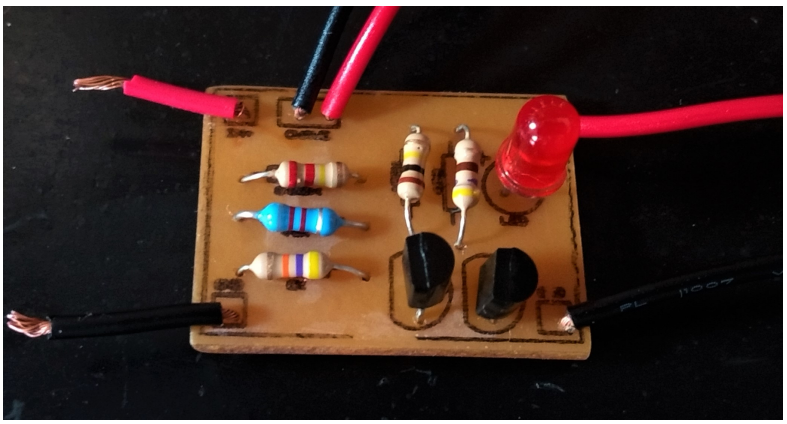

Figura 3.8 – Placa desenvolvida para indicar carga baixa da bateria.

Com isso, o circuito de alimentação completo do módulo consiste na bateria 18650 conectada ao módulo de carga TP4056 seguido de uma chave *on-off* responsável por ativar ou desativar todo o dispositivo. A chave está conectada ao circuito indicador de carga da bateria e na sequência tem-se o conversor DC boost step up alimentando o módulo com os componentes.

3.2.2 Desenvolvimento do Dispositivo

Após definir o posicionamento de cada componente no módulo partiu-se para a definição dos pinos em que cada componente seria conectado. Essa escolha, levou em conta as necessidades de comunicação de cada componente e as características de cada pino do ESP-32. A pinagem final pode ser vista na Tabela 3.4.

Tabela 3.4 – Tabela de pinagem do ESP-32 com os componentes.

Componente	**Pino do componente**	**Pino do ESP-32**
Ultrassom HC-SR04	Trig	5
	Echo	35
Display 7 Segmentos	CLK	18
	MOSI	23
Botões *Touch*	1	14
	2	27
	3	33
PushButton	Esquerdo	36
	Direito	16
LED RGB	Vermelho	0
	Verde	2
	Azul	15
Buzzer	Positivo	4
Sensor de temperatura DHT11	Sinal	13
Servo Motor	Sinal	17
LEDs	Azul	22
	Verde	25
	Amarelo	26
	Vermelho	32
Potenciômetro	Sinal	34
LDR	Sinal	39

Vale ressaltar que existem algumas peculiaridades nos pinos do ESP32, e todas elas foram consideradas. Dentre as peculiaridades pode-se citar os pinos GPIO 34, 35, 36 e 39 que são apenas de entrada, o pino GPIO 02 é conectado ao LED da placa e os pinos GPIO 1 e 3 são para comunicação serial (TX e RX) e por isso, estes dois últimos, foram deixados disponíveis. Outros pinos que não foram utilizados foram conectados a barras de pinos fêmea para que o usuário possa usá-los, caso queira utilizar componentes extras.

Finalizando a etapa de testes, realizou-se todas as conexões utilizando uma *protoboard* e testou-se o funcionamento de cada componente, buscando validar o desempenho dos mesmos. Foi implementado códigos simples e genéricos que podem ser vistos no Apêndice D. E na Figura 3.9 é apresentado a montagem realizada para execução dos testes. Outro registro dos testes sendo executados pode ser visto no Apêndice B, que lista alguns *links* com vídeos de gravações dos testes.

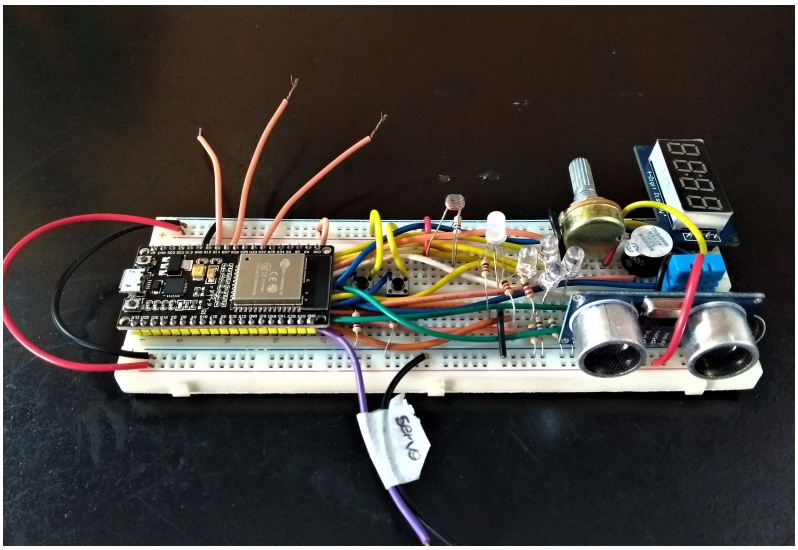

Figura 3.9 – Foto do dispositivo em fase de testes.

Partindo para a etapa final do desenvolvimento do módulo tem-se a construção da placa de circuito impresso. Foi utilizado o *software* PROTEUS para construção do esquemático, apresentado no Apêndice C, e todo o circuito a ser impresso e transferido para a placa.

A construção da placa se deu pelo mesmo método utilizado na construção da placa de circuito indicador de carga, apresentado anteriormente, uma alteração neste caso se deu pela necessidade de utilizar uma placa de fenolite dupla face. Na Figura 3.10 abaixo pode-se observar a placa desenvolvida em conjunto com o circuito de alimentação. Além disso, pode-se observar o perfeito funcionamento do módulo completo e finalizado através do vídeo listado no Apêndice B.

3.2.3 Projeto da Case

Buscando tornar o produto visualmente mais atrativo e de fácil manuseio foi elaborado um projeto de case. Com isso, definiu-se que os botões *touch* ficaria na parte inferior e o sensor ultrassônico na parte superior.

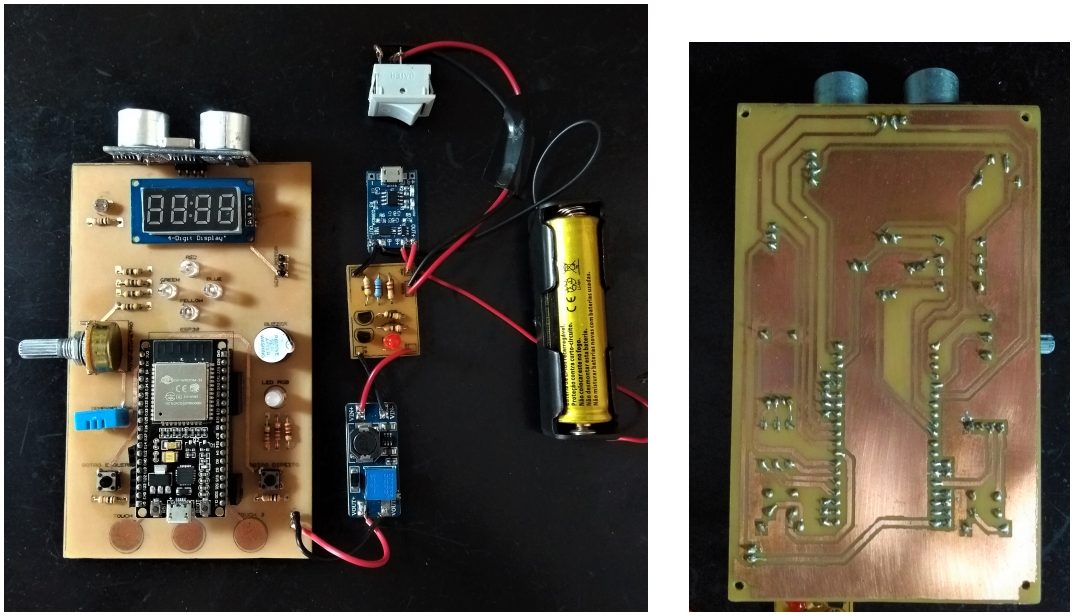

Figura 3.10 – Projeto final do dispositivo com o circuito de alimentação.

Além disso, na lateral do módulo foi estruturado um compartimento para alocar todo o circuito de alimentação com a bateria. Nesta região se encontra a chave *on-off*, LED indicador de carga baixa da bateria e dois LEDs estendidos do módulo TP4056 para indicar, quando a bateria estiver sendo recarregada, se a carga está completa ou se a recarga ainda está sendo feita.

Na Figura 3.11 é possível ver o projeto final da case. E sua dimensão é 116x136x40 em milímetros, com 3mm de espessura da parede.

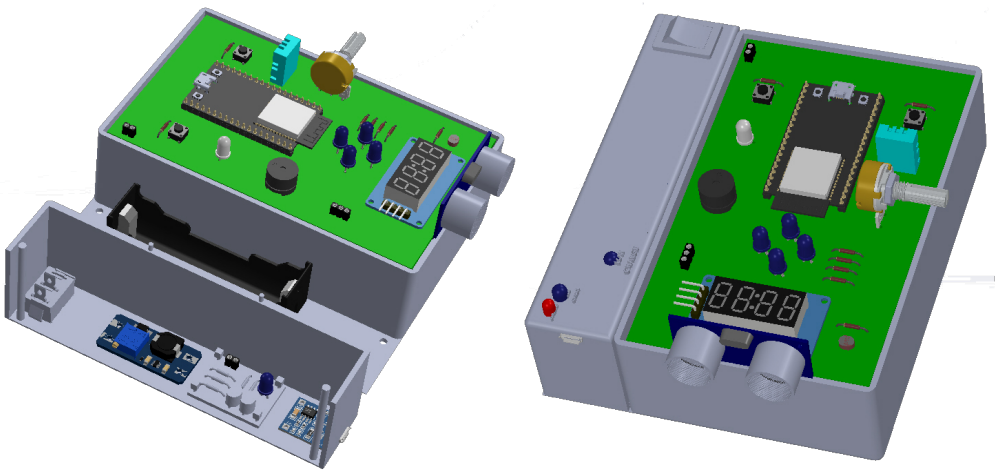

Figura 3.11 – Desenho da case com todos os componentes.

Definiu-se também que a melhor forma de se construir a case seria com a produção de duas peças distintas, uma representando a base e outra o compartimento lateral.

3.3 O Aplicativo

Como primeiro passo, definiu-se as funcionalidades do aplicativo. O mesmo deve ser prático, intuitivo e objetivo. O aplicativo possui uma tela inicial, tela para carregamento de projetos da biblioteca, uma tela para criação/programação de novos códigos e uma área contendo uma série de conteúdos para estudo.

Tela do menu - Primeira tela do aplicativo. Nela, estará a logo do aplicativo e o usuário deverá escolher entre 4 botões.

Tela para novo arquivo - Nessa tela, o usuário terá a possibilidade de criar seu próprio código, compilar, identificar possíveis problemas no código e carregar o programa para o microcontrolador.

Tela para carregar arquivo - O intuito dessa tela é permitir que o usuário carregue para o microcontrolador um código já pronto, presente dentro da biblioteca do aplicativo

Banco de conteúdos - Nesta área do aplicativo o usuário terá contato com diversos conteúdos, que possibilitarão o estudo das estruturas clássicas de programação.

Banco de Projetos - Nessa área, o usuário terá contato com conteúdos explicando a parte eletrônica e o funcionamento de componentes do módulo. Além disso, ele também conseguirá visualizar uma série de projetos exemplos para se inspirar e se basear na criação do seu próprio projeto.

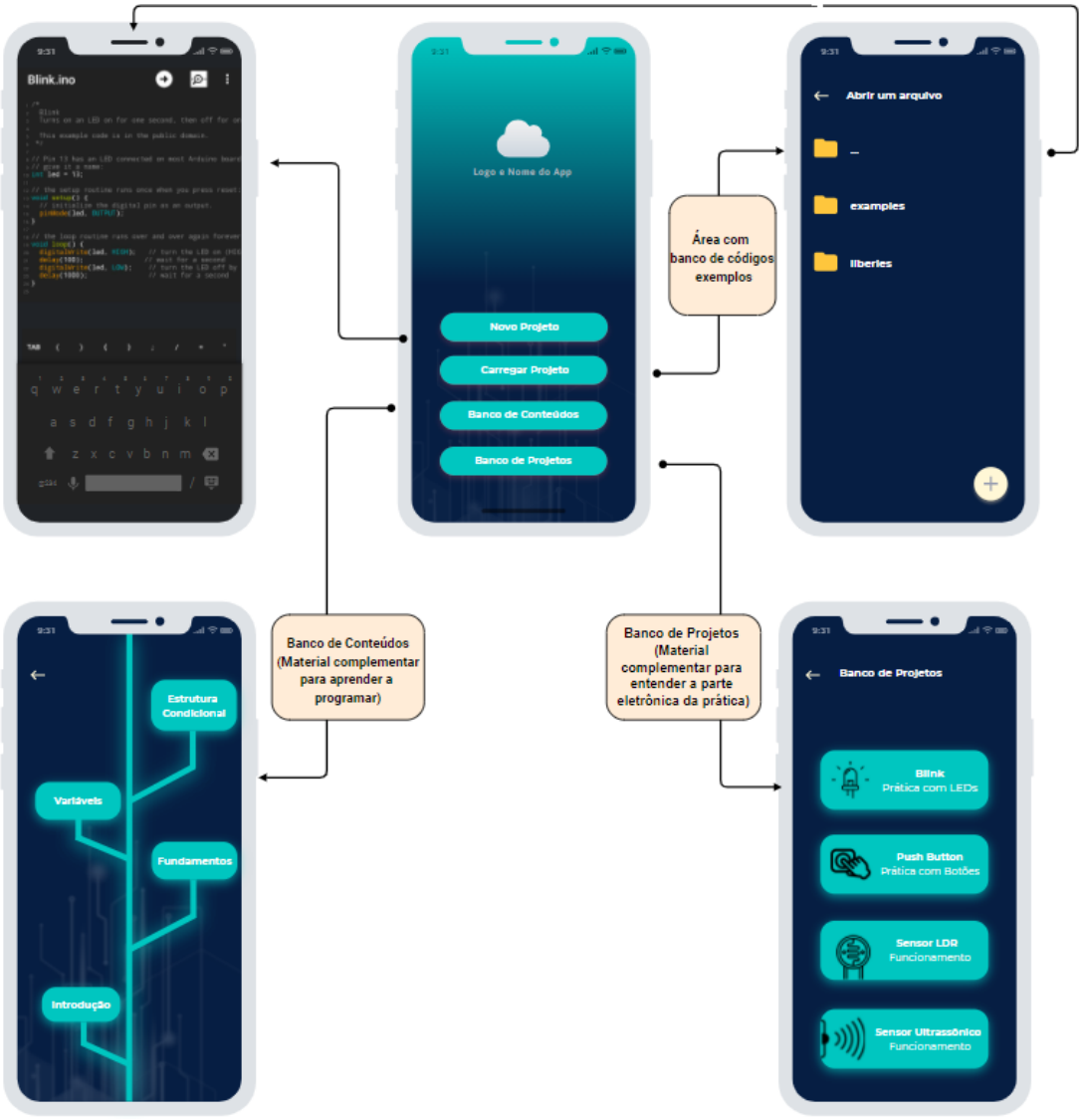

Figura 3.12 – Esboço das telas do Aplicativo. Fonte: Elaborado pelo autor.

3.4 Procedimento de Uso

A fim de exemplificar o procedimento de uso foi elaborado um Diagrama de caso de uso UML. O objetivo do diagrama de caso de uso em UML é demonstrar, de forma geral, as diferentes maneiras que o usuário pode interagir com um sistema.

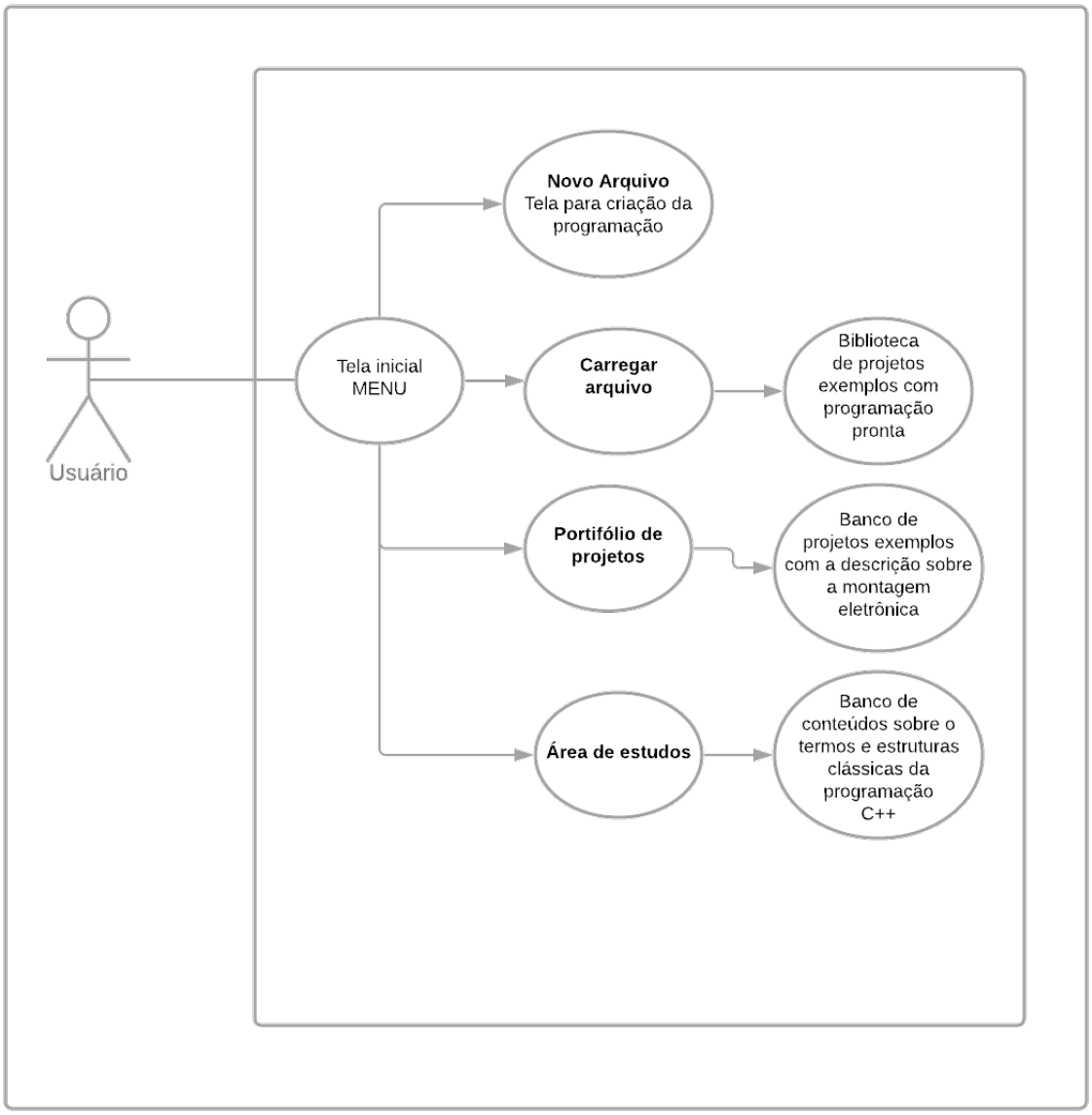

Figura 3.13 – Diagrama de caso de uso.Fonte: Elaborado pelo autor.

3.5 Desenvolvimento do Aplicativo

Com as etapas descritas anteriormente concluídas pariu-se para o desenvolvimento do aplicativo de fato. Durante todo o desenvolvimento da programação objetivou um layout e uma usabilidade mais simples, lúdica e intuitiva. Para que usuários de diferentes idades conseguissem utilizar o aplicativo.

Para garantir maior embasamento nesta etapa, foi consultado durante todo o processo a documentação no site oficial do *React Native*. Além disso, também foi adquirido e realizado um curso online.

A primeira página criada foi a tela de menu inicial. Foi adicionado a logo com o nome

do aplicativo e os quatro botões direcionando, respectivamente, para a área de editor de códigos, área com banco de projetos exemplos, área com banco de conteúdos sobre programação e por último para a área com um banco de conteúdos sobre eletrônica básica e robótica. Vale ressaltar que as áreas de editor de código e de banco de programas exemplos não foram desenvolvidas neste momento, pois não são objetivos deste trabalho. Abaixo, na Figura 3.14, é possível observar a tela de menu criada.

Figura 3.14 – Tela do menu inicial .Fonte: Elaborado pelo autor.

Para a navegação entre telas foi utilizado as bibliotecas do *React Native Navigation* e foi utilizado o método de *Stacks* (pilhas). O navegador de pilha nativo do *React Navigation* fornece uma maneira do aplicativo fazer a transição entre as telas e gerenciar o histórico de navegação. Além disso é possível adicionar animações e mecânicas para deixar a utilização do aplicativo mais fluida e atrativa.

A seguir, na Figura 3.15 é apresentado as telas da área do aplicativo que contém o banco de conteúdos sobre programação. Foi adicionado diversos botões, que direcionam para diferentes páginas com diferentes conteúdos. E para que todos os botões pudessem ser acessados pelo usuário foi adicionado a mecânica de *scroll*. Além disso, foi adiconado um cabeçalho com uma animação, que ao usuário realizar o *scroll* da tela o cabeçalho diminui de tamanho até desaparecer, voltando somente quando o usuário descer a tela.

Para a tela contendo o banco de conteúdos sobre eletrônica básica e robótica foi realizado uma programação ligeiramente semelhante da tela apresentada anteriormente,

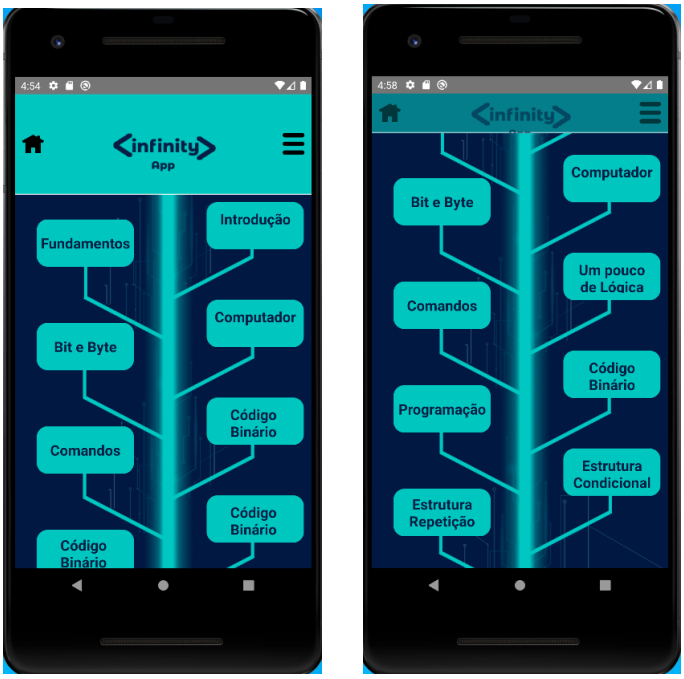

Figura 3.15 – Tela do banco de conteúdos sobre programação .Fonte: Elaborado pelo autor.

com a mecânica de *scroll* e com a animação do cabeçalho. Porém, neste caso a disposição dos botões foi diferente e foi adicionado a cada botão um símbolo do componente referente ao conteúdo, como pode ser visto na Figura 3.16 abaixo.

Figura 3.16 – Tela do banco de conteúdos sobre eletrônica e robótica .Fonte: Elaborado pelo autor.

Na sequência está disposto quatro imagens, referentes as Figuras 3.17 e 3.18, que

demonstram a disposição dos conteúdos para aprendizado. Lembrando que para acessar as telas de conteúdo basta clicar no botão referente ao conteúdo que deseja estudar.

Para ter acesso a todo código desenvolvido na criação do aplicativo, o mesmo foi adicionado à um repositório do *GitHub* e pode ser acessado através do link presente no Anexo E.

Figura 3.17 – Exemplo de Tela do banco de conteúdos.Fonte: Elaborado pelo autor.

Após a finalização da primeira versão do aplicativo, executando as etapas descritas anteriormente, foi realizado uma atualização. Foi adicionado outra biblioteca do *React Navigation*, a biblioteca de navegação *Tab*. Tal biblioteca permite a navegação entre telas através de um rodapé adicionado na tela desejada, com alguns botões.

Na Figura 3.19, pode ser visto a aplicação de tal funcionalidade, que foi adicionada visando maior facilidade de acesso entre as telas, por parte do usuário. Portanto, foi adicionado em todas as telas, contendo conteúdos sobre programação e eletrônica, uma barra na parte inferior da tela. Nesta barra está disposto botões que podem direcionar para a tela de menu inicial, para a tela de banco de exemplos ou para a tela de editor de código para criar um novo projeto após o estudo do conteúdo.

Figura 3.18 – Exemplo de Tela do banco de conteúdos sobre eletrônica e robótica .Fonte: Elaborado pelo autor.

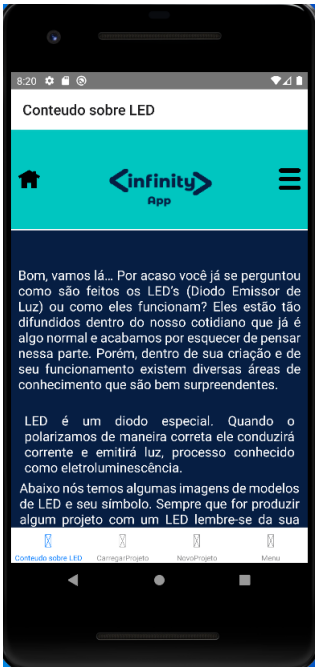

Figura 3.19 – Atualização de Tela de conteúdo com a borda de botões.Fonte: Elaborado pelo autor.

3.6 Investimentos do Projeto

A Tabela 3.5 a seguir apresenta a estimativa de investimentos para construção do módulo em uma coluna e na coluna mais a direita a observação de investimento que foi necessário para a elaboração deste trabalho.

Tabela 3.5 – Tabela de investimentos (orçamento).

Produto	Investimento (R$)	Observação
Eletrônica *ESP 32*	49,49	Adquirido
Módulo de carga TP4065	3,30	Adquirido
Conversor DC Boost Step Up	7,15	Adquirido
Sensor de Umidade e Temperatura DHT11	11,55	Adquirido
LEDs	0,85	Já possuído
Push Button (Chave Táctil)	0,35	Já possuído
Sensor LDR	0,50	Já possuído
Buzzer ativo 5mm	2,50	Já possuído
Sensor Ultrassónico HC-SR04	11,00	Já possuído
Display 7 segmentos MT1637	11,40	Adquirido
Potenciômetro 1kΩ	2,00	Já possuído
Suporte bateria 18650	7,90	Adquirido
Servo Motor SG90	11,50	Já possuído
Placa de Fenolite dupla face	8,00	Adquirido
Cabos e outros componentes	5,00	Já possuído
Aplicativo Celular com sistema operacional Android	-	Já possuído
Total	132,49	R$98,79

Os recursos utilizados para aquisição dos componentes foram do próprio do autor.

Tal investimento apresentado se justifica, pois se comparado com as tecnologias atuais que existem no mercado é um valor muito inferior ao que se pratica. Demonstrando que o módulo referente a este trabalho é uma alternativa mais acessível para o ensino e prática de programação e robótica.

Capítulo 4

Considerações Finais

Neste capítulo as principais observações concluídas no decorrer do desenvolvimento do TCC são apresentadas, bem como as propostas de continuidade.

4.1 Conclusões

O presente trabalho teve como objetivo o projeto de um módulo didático para o ensino de programação, sendo este constituído de duas partes, uma placa com o microcontrolador, sensores e atuadores e a segunda parte sendo um aplicativo, atuando como um ambiente para estudo e prática de programação e robótica. O primeiro passo para alcançar tal fim, foi a realização de pesquisas acerca do tema, buscando entender sobre as necessidades de projeto, os projetos existentes e evidenciando a importância do dispositivo a ser desenvolvido.

Feito isso, os componentes a serem empregados no projeto foram definidos e partiu-se para uma breve revisão bibliográfica, evidenciando o grande avanço tecnológico dos componentes envolvidos. Também nesse ponto, percebeu-se que existem muitos projetos e estudos envolvendo kits para o ensino e prática de programação, mas poucos que se comprometeram a desenvolver algo de baixo custo e com a utilização de um aplicativo como ferramenta.

Posteriormente, com todas as informações obtidas na produção do referencial teórico, pode-se realizar o desenvolvimento. Foi estruturado o layout e esboço para o aplicativo e as funcionalidades que o mesmo teria e, devido a variedade de opções de microcontroladores, encontrou-se dificuldades para realizar testes com todos e validar o mais indicado.

Ainda como parte do desenvolvimento deste trabalho foi selecionado o microcontrolador e componentes que seriam integrados no módulo. Feito isto, foi aplicado diversos testes da integração do microcontrolador com os componentes, foi calculado a necessidade que o dispositivo teria. Por fim, foi dimensionado o sistema de alimentação e construído a placa.

Em paralelo as etapas descritas anteriormente, foi realizado o desenvolvimento do aplicativo. Construindo as páginas contendo conteúdos para o aprendizado em programação, eletrônica e robótica.

Por fim, tem-se que o projeto foi desenvolvido de acordo com o cronograma proposto e alcançou resultados satisfatórios e interessantes, principalmente devido ao baixo custo de investimento. Além disso, o trabalho permitiu a utilização das capacidades e conhecimentos obtidos durante o curso de Engenharia Mecatrônica com a interdisciplinaridade das áreas.

4.2 Propostas de Continuidade

De acordo com as percepções e dificuldades encontradas no decorrer deste trabalho sugere-se como propostas de continuidade os itens a seguir:

- Desenvolvimento da segunda versão do aplicativo. Pois com essa versão será possível o usuário ter acesso a página de editor de código, para assim, criar e compilar seus próprios códigos;

- Desenvolvimento da terceira versão do aplicativo e consequente implementação da comunicação sem fio entre o aplicativo e o microcontrolador no processo de carregamento do código. Pois, assim, o usuário não irá necessitar de um celular com saída micro-USB OTG para transferir o código por cabo para o microcontrolador;

- Implementação de novos conteúdos no aplicativo para o estudo sobre programação e eletrônica básica;

- Implementação de práticas com programação em blocos para o processo de aprendizado. Pois, assim, o aplicativo irá fornecer um recurso a mais para crianças aprenderem programação;

- Implementação de uma tela de login com autenticação com contas Google ou Microsoft, para possibilitar o armazenamento na nuvem do código criado.

Apêndice A

Circuito Indicador de Carga Baixa

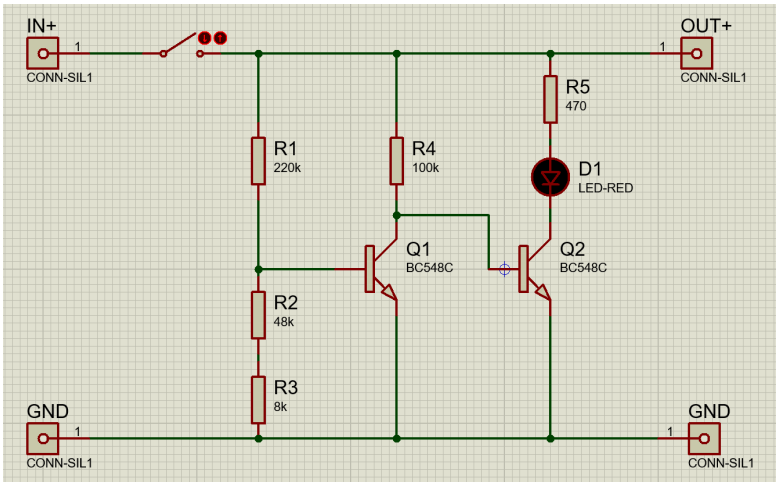

Figura A.1 – Circuito desenvolvido no PROTEUS.

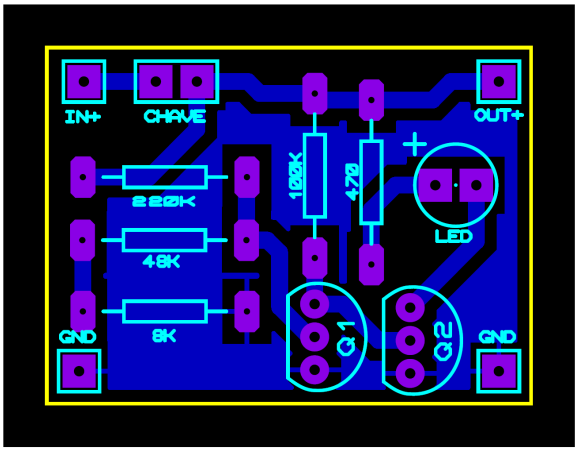

Figura A.2 – Layout da PCB

Apêndice B

Vídeos do Funcionamento do Projeto

B.1 Projeto Final

O vídeo demonstrando o funcionamento do circuito de alimentação e o módulo em seu projeto final pode ser visto no seguinte link: `https://youtu.be/M2nx4eN8zk4`.

B.2 Teste com componentes do módulo

Vídeo demonstrando o funcionamento dos componentes: LEDs, LED RGB, LDR, Potenciômetro, *Push Buttons* e Botões *Touch*. O vídeo da fase de testes pode ser visto no seguinte link: `https://youtu.be/MNgf2L8mqro`.

B.3 Teste com Servo Motor

Vídeo demonstrando o funcionamento do servo motor em fase de testes pode ser visto no seguinte link: `https://youtu.be/vmPahE-8duA`.

B.4 Teste com Sensor Ultrassom

Vídeo demonstrando o funcionamento do sensor ultrassom, display de 7 segmentos e buzzer em fase de testes pode ser visto no seguinte link: `https://youtu.be/M08np18CWhc`.

B.5 Teste com Sensor de Temperatura

Vídeo demonstrando o funcionamento do sensor de temperatura e umidade DHT11 e o display de 7 segmentos em fase de testes pode ser visto no seguinte link: `https://youtu.be/WrK__NbLr7g`.

Apêndice C

Esquemático do módulo

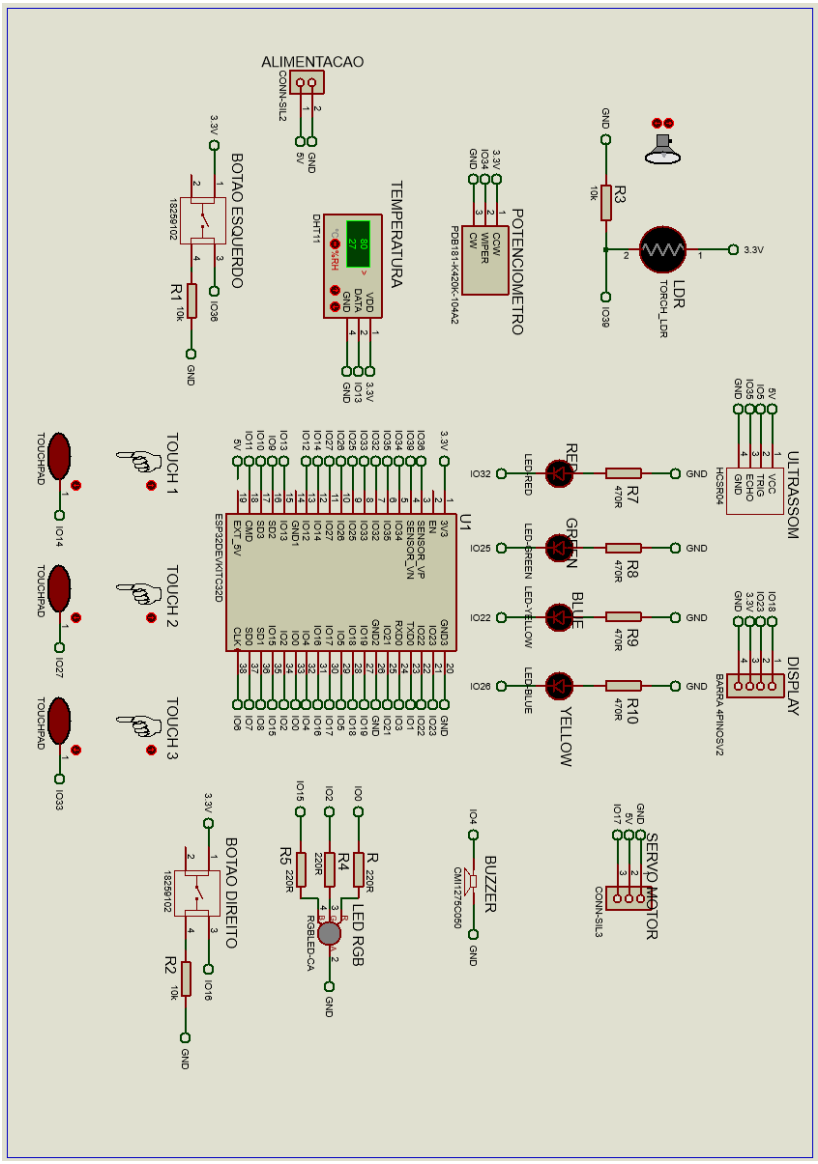

Figura C.1 – Circuito esquemático do módulo desenvolvido no PROTEUS.

Apêndice D

Código de Testes do Microcontrolador

D.1 Código para Teste dos componentes

```
1  //Sensor Touch
2  int pinoTouch1 = 14; // GIOP14
3  int pinoTouch2 = 27; // GIOP27
4  int pinoTouch3 = 33; // GIOP33
5  int capacitanciaMaxima = 40; //valor que nos da a certeza de toque
6
7  // LED RGB
8  int PIN_RED = 0; // GIOP0 - ADC11
9  int PIN_GREEN = 2; // GIOP2 - ADC12
10 int PIN_BLUE =  15; // GIOP15 - ADC13
11 const int channelR = 11;
12 const int channelG = 12;
13 const int channelB = 13;
14 const int frequency = 1000;
15 const int resolution = 8;
16
17 //PINOS DOS BOTÕES
18 int buttonESQ = 36; // GIOP36 - ADC0 - Somente entrada
19 int buttonDIR = 16; // GIOP16
20 bool estado1 = false;
21 bool estado2 = false;
```

```
22
23 //PINOS DOS LEDs
24 int led_amarelo = 26; // GIOP26 - ADC19
25 int led_verde = 25;   // GIOP25 - ADC18
26 int led_vermelho = 32;// GIOP32 - ADC4
27 int led_azul = 22;    //
28 const int channelLEDvermelho = 10;
29
30 const int potenciometro = 34; // GIOP34 - ADC6 - Somente entrada
31 const int LDR = 39;           // GIOP39 - ADC3 - Somente entrada
32
33 void setup(){
34
35   //SETA OS PINOS DOS PUSH BUTTONS COMO ENTRADAS
36   pinMode(buttonESQ, INPUT);
37   pinMode(buttonDIR, INPUT);
38
39   //SETA OS PINOS DOS LEDs COMO SAÍDA
40   pinMode(led_verde, OUTPUT);
41   pinMode(led_amarelo, OUTPUT);
42   pinMode(led_azul, OUTPUT);
43   //LED VERMELHO
44   ledcSetup(channelLEDvermelho,frequency,resolution);
45   ledcAttachPin(led_vermelho,channelLEDvermelho);
46
47   //SETA OS PINOS DOS LEDs RGB COMO SAÍDA ANALÓGICA
48   ledcSetup(channelR,frequency,resolution);
49   ledcAttachPin(PIN_RED,channelR);
50   ledcSetup(channelG,frequency,resolution);
51   ledcAttachPin(PIN_GREEN,channelG);
52   ledcSetup(channelB,frequency,resolution);
53   ledcAttachPin(PIN_BLUE,channelB);
54
55   pinMode(potenciometro, INPUT);
```

```
56    pinMode(LDR, INPUT);
57    //Abre a porta serial, definindo a taxa de dados
58    Serial.begin(115200);
59  }
60  void loop(){
61
62    //CODIGO DO SENSOR TOUCH COM LED RGB
63    int media1 = 0;
64    int media2 = 0;
65    int media3 = 0;
66    //faz 100 leituras de cada sensor touch e calcula a média do valor
        lido
67    for(int i=0; i< 100; i++){
68      media1 += touchRead(pinoTouch1);
69      media2 += touchRead(pinoTouch2);
70      media3 += touchRead(pinoTouch3);
71    }
72
73    media1 = media1 / 100;
74    media2 = media2 / 100;
75    media3 = media3 / 100;
76
77    //verifica se o valor médio lido no pino é menor que a capacitância
        máxima definida
78    //se verdadeiro, isso caracteriza um toque
79    if(media1 < capacitanciaMaxima){
80       // color code #00C9CC (R = 0,   G = 201, B = 204)
81       ledcWrite(channelR,0);
82       ledcWrite(channelG,201);
83       ledcWrite(channelB,204);
84    }
85    else if(media2 < capacitanciaMaxima){
86      // color code #F7788A (R = 247, G = 120, B = 138)
87      ledcWrite(channelR,247);
```

```
88      ledcWrite(channelG, 120);
89      ledcWrite(channelB,  138);
90
91    }else if(media3 < capacitanciaMaxima){
92      // color code #34A853 (R = 52,  G = 168, B = 83)
93      ledcWrite(channelR,   52);
94      ledcWrite(channelG, 168);
95      ledcWrite(channelB,  83);
96    }
97    //se nenhum dos pinos touch estão sendo tocados, os LEDS ficam
         apagados
98    else{
99      ledcWrite(channelR, 0);
100     ledcWrite(channelG, 0);
101     ledcWrite(channelB, 0);
102   }
103   delay(10);
104
105 //CODIGO POTENCIOMETRO
106   int potencia = analogRead(potenciometro);
107   Serial.println(potencia);
108   potencia = 4095 - potencia;
109
110   //atribui o valor lido do potenciômetro para configurar a
         intensidade do brilho do LED
111   ledcWrite(channelLEDvermelho, potencia);
112   Serial.println(potencia);
113
114 //CÓDIGO DOS PUSH BUTTONS COM LEDS AMARELO E AZUL
115   if(estado1 == true){
116     //Ligamos o LED Amarelo
117     digitalWrite(led_amarelo, HIGH);
118   }else{
119     digitalWrite(led_amarelo, LOW);
```

```
120   }
121   if(estado2 == true){
122     //Ligamos o LED Azul
123     digitalWrite(led_azul, HIGH);
124   }else{
125     digitalWrite(led_azul, LOW);
126   }
127   if (digitalRead(buttonESQ) == HIGH){ //Caso o botão 1 foi
        pressionado
128     estado1 = !estado1;
129   }
130   if (digitalRead(buttonDIR) == HIGH){ //Caso o botão 2 foi
        pressionado
131     estado2 = !estado2;
132   }
133
134  //CODIGO LDR
135  int val = analogRead(LDR);
136  if(val<1000){
137   digitalWrite(led_verde, HIGH);
138  }else{
139   digitalWrite(led_verde, LOW);
140   }
141 }
```

Listing D.1 – Código do microcontrolador

D.2 Código para Teste do Servo Motor

```
1 //SERVO MOTOR
2 #include <ESP32Servo.h>
3 Servo myservo;  // cria o objeto servo
4 int pos = 0;
5 int servoPin = 17; // GIOP17
6
```

```
7  void setup(){
8
9    //CODIGO PARA SERVO MOTOR
10   // Allow allocation of all timers
11   ESP32PWM::allocateTimer(0);
12   ESP32PWM::allocateTimer(1);
13   ESP32PWM::allocateTimer(2);
14   ESP32PWM::allocateTimer(3);
15   myservo.setPeriodHertz(50);    // standard 50 hz servo
16   myservo.attach(servoPin, 1000, 2000); // attaches the servo on pin
        18 to the servo object
17
18   //Abre a porta serial, definindo a taxa de dados
19   Serial.begin(115200);
20 }
21 void loop(){
22   //CODIGO DO SERVO MOTOR
23   for (pos = 0; pos <= 180; pos += 1) { // goes from 0 degrees to 180
        degrees
24     // in steps of 1 degree
25     myservo.write(pos);    // tell servo to go to position in
          variable 'pos'
26     delay(15);             // waits 15ms for the servo to reach the
          position
27   }
28   for (pos = 180; pos >= 0; pos -= 1) { // goes from 180 degrees to 0
        degrees
29     myservo.write(pos);    // tell servo to go to position in
          variable 'pos'
30     delay(15);             // waits 15ms for the servo to reach the
          position
31   }
32 }
```

Listing D.2 – Código do microcontrolador

D.3 Código para Teste do Sensor Ultrassom

```
1  //Biblioteca responsável para comunicação com o display de 7
       segmentos
2  #include <TM1637Display.h>
3
4  //Carrega a biblioteca do sensor ultrassonico
5  #include <Ultrasonic.h>
6
7  #define INTERVALO_LEITURA 250 //(ms)
8
9  //conexão dos pinos para o sensor ultrasonico
10 #define PIN_TRIGGER   5
11 #define PIN_ECHO      35
12
13 // Module TM1637 pins
14 #define CLK 18
15 #define DIO 23
16
17 //Inicializa o display nos pinos definidos acima
18 TM1637Display display(CLK, DIO);
19
20 //definição dos PINOS que utilizaremos para os LEDS e o BUZZER
21 #define PIN_BLUE_LED    22
22 #define PIN_GREEN_LED   32
23 #define PIN_YELLOW_LED  25
24 #define PIN_RED_LED     26
25 #define PIN_BUZZER      4
26
27 unsigned int distancia = 0;
28
29 //Inicializa o sensor nos pinos definidos acima
30 Ultrasonic ultrasonic(PIN_TRIGGER, PIN_ECHO);
31
32 void setup(){
```

```
33    Serial.begin(9600);
34    configurarPinos();
35    //configura o brilho do display com valor máximo
36    display.setBrightness(0x0a);
37  }
38  void loop(){
39    verificarDistancia();
40    delay(INTERVALO_LEITURA);
41  }
42  //CONFIGURA O MODOS DE CADA UM DOS PINOS QUE UTILIZAREMOS COMO SAIDA
43  void configurarPinos(){
44    pinMode(PIN_BLUE_LED,   OUTPUT);
45    pinMode(PIN_GREEN_LED,  OUTPUT);
46    pinMode(PIN_YELLOW_LED, OUTPUT);
47    pinMode(PIN_RED_LED,    OUTPUT);
48    pinMode(PIN_BUZZER,     OUTPUT);
49  }
50  /*
51    VERIFICA A DISTANCIA ATUAL QUE O SENSOR ULTRASONICO ESTA LENDO
52    E EM SEGUIDA, IMPRIME O VALOR NO DISPLAY, E ACENDE O LED
        CORRESPONDENTE
53  */
54  void verificarDistancia(){
55
56    //recupera a distância atual lida pelo sensor
57    distancia = getDistance();
58
59    //imprime no display o valor lido
60    display.showNumberDec(distancia);
61
62    Serial.print("Distancia: ");
63    Serial.println(distancia);
64
65    //esse FOR tem como objetivo apagar todos os LEDS que estejam
```

```
    acesos.
  for(int i=PIN_BLUE_LED; i<=PIN_RED_LED; i++)  {
    digitalWrite(i, LOW);
  }
  //desliga o BUZZER
  digitalWrite(PIN_BUZZER, LOW);

  //caso a distancia lida seja menor ou igual a 5, tomaremos como uma
      distância de perigo
  //então acenderemos o LED VERMELHO e ligaremos o BUZZER
  if( distancia <= 5 )  {
    digitalWrite(PIN_RED_LED, HIGH);
    digitalWrite(PIN_BUZZER, HIGH);
  }
  //caso a distancia seja maior que 5 e menor ou igual a 20,
  //tomaremos como uma distância de atenção, e ligaremos o LED
     AMARELO
  else if(distancia <=20)  {
    digitalWrite(PIN_YELLOW_LED, HIGH);
  }
  //caso a distancia seja maior que 20 e menor ou igual a 40,
  //tomaremos como uma distância segura, e ligaremos o LED VERDE
  else if(distancia <= 40){
    digitalWrite(PIN_GREEN_LED, HIGH);
  }
  //para distâncias maiores que 40, tomaremos como uma distância sem
     perigo
  //acenderemos o LED AZUL para indicar
  else  {
    digitalWrite(PIN_BLUE_LED, HIGH);
  }
}
//FAZ A LEITURA DA DISTANCIA ATUAL CALCULADA PELO SENSOR
int getDistance(){
```

```
96      //faz a leitura das informacoes do sensor (em cm)
97      int distanciaCM;
98      long microsec = ultrasonic.timing();
99      distanciaCM = ultrasonic.convert(microsec, Ultrasonic::CM);
100
101     return distanciaCM;
102 }
```

Listing D.3 – Código do microcontrolador

D.4 Código para Teste do Sensor de Temperatura

```
1  //Biblioteca responsável para comunicação com o display de 7
       segmentos
2  #include <TM1637Display.h>
3
4  //Carrega a biblioteca do sensor DHT11
5  #include <SimpleDHT.h>
6  #define DHTPIN 13
7
8  //Intervalo entre cada leitura do sensor
9  #define INTERVAL 5000
10
11 //Objeto que realiza a leitura da umidade e temperatura
12 SimpleDHT11 dht;
13
14 //Variáveis que vão guardar o valor da temperatura e umidade
15 float temperature, humidity;
16
17 //Marca quando foi feita a última leitura
18 uint32_t lastTimeRead = 0;
19
20 // Module connection pins (Digital Pins)
21 #define CLK 18
22 #define DIO 23
```

```
23
24 //Inicializa o display nos pinos definidos acima
25 TM1637Display display(CLK, DIO);
26
27 void setup(){
28   Serial.begin(115200);
29   //configura o brilho do display com valor máximo
30   display.setBrightness(0x0a);
31 }
32
33 void loop(){
34  //Tempo em millissegundos desde o boot do esp
35  unsigned long now = millis();
36
37  //Se passou o intervalo desde a última leitura
38  if(now - lastTimeRead > INTERVAL){
39    //Faz a leitura do sensor
40    float t, h;
41    //Coloca o valor lido da temperatura em t e da umidade em h
42    int status = dht.read2(DHTPIN, &t, &h, NULL);
43    //Se a leitura foi bem sucedida
44    if (status == SimpleDHTErrSuccess) {
45      //Os valores foram lidos corretamente, então é seguro colocar
           nas variáveis
46      temperature = t;
47      humidity = h;
48    }
49    //Mostra no display
50    delay(500);
51    display.showNumberDec(temperature);
52    delay(1000);
53    display.showNumberDec(humidity);
54    delay(500);
55    //Marca quando ocorreu a última leitura
```

```
56     lastTimeRead = now;
57
58    Serial.print("Temperatura: ");
59    Serial.println(temperature);
60    Serial.print("Umidade: ");
61    Serial.println(humidity);
62   }
63 }
```

Listing D.4 – Código do microcontrolador

Apêndice E

Código do Aplicativo

O código completo do aplicativo pode ser visto no seguinte link: `https://github.com/MartinsMC/AppInfinity_TCC`.

Referências

ARAUJO, G. R. de. Desenvolvimento cross-platform com react native: um estudo de caso do aplicativo Naveg. Ceará, 2019.

ARDUINO. **Arduino Products**. 2021. Disponível em: <https://www.arduino.cc/>. Acesso em: 13 mai. 2021.

AYCOCK, S. **A história dos microcontroladores**. TED, 2017. Disponível em: <https://www.ehow.com.br/controlador-smbus-fatos_94209/>. Acesso em: 13 mai. 2021.

BASSANI, J. V. Desenvolvimento de um kit de robótica de baixo custo aplicado ao ensino no IFSC. Florianópolis, 2020.

BECKER, L. **O que é React Native**. Disponível em: <https://www.organicadigital.com/blog/o-que-e-react-native/>. Acesso em: 3 jul. 2021.

BOLTON, W. **Mecatrônica: uma abordagem multidisciplinar**. 4. ed. Porto Alegre: Bookman, 2010. ISBN 978-85-7780-657-7.

BUSSON, R. S. Desenvolvimento de uma plataforma educacional de apoio ao ensino e aprendizagem de robótica à luz da pedagogia de projetos. Porto Velho, 2017.

CASTRO, A.; GIMENES, I. M.; C. FILHO, J. A. de. Computação na Educação Básica: Breve Contextualização Histórica. Computação Brasil, p. 30–34, 2019.

CASTRO, V. G. de. RoboEduc: especificação de um software educacional para ensino da robótica às crianças como uma ferramenta de inclusão digital. Natal, 2008.

CODE8734. **Fluência Computacional**. Editora CODE8734. Disponível em: <https://www.code8734.com.br/blog/fluencia-computacional>. Acesso em: 16 jun. 2021.

COELHO, I. **Qual módulo WiFi ESP32 é ideal para meu projeto?** Filipeflop. Disponível em: <https://www.filipeflop.com/blog/qual-modulo-wifi-esp32-e-ideal-para-meu-projeto/>. Acesso em: 18 jun. 2021.

ESPRESSIF. **ESP32-WROOM-32E e ESP32-WROOM-32UE Datasheet**. 1.2. ed. Espressif Systems. Disponível em: <https://www.espressif.com/sites/default/files/documentation/esp32-wroom-32e_esp32-wroom-32ue_datasheet_en.pdf>. Acesso em: 18 jun. 2021.

EVANS, M.; NOBLE, J.; HOCHENBAUM, J. **Arduino em Ação**. 1. ed. São Paulo: Novatec, 2013. ISBN 978-85-7522-373-4.

FOGAÇA, R. A. O. Desenvolvimento de um software educacional gamificado para plataforma web com ferramenta de autoria de questões. Paraná, 2018. Disponível em: <http://repositorio.roca.utfpr.edu.br/jspui/handle/1/10420>.

GOLDMAN, R.; EGUCHI, A.; SKLAR, E. Using education robotics to engage inner-city students with technology. Proceedings of the 6th international conference on Learning sciences, Santa Monica, California, p. 214–221, 2004.

GOOGLE. **Todos os alunos devem ter a chance de aprender ciência da computação**. Google For Education, 2017. Disponível em: <https://edu.google.com/intl/ALL_br/computer-science/>. Acesso em: 21 mai. 2021.

IBGE, Educa. **Pesquisa sobre o uso das tecnologias de informação e comunicação no Brasil: TIC Domicílios e Empresas 2019**. Comitê Gestor da Internet no Brasil, 2019. Disponível em: <https://educa.ibge.gov.br/jovens/materias-especiais/20787-uso-de-internet-televisao-e-celular-no-brasil.html#subtitulo-0>. Acesso em: 10 mai. 2021.

JUNIOR, A. O. C.; GUEDES, E. B. Uma Análise Comparativa de Kits para a Robótica Educacional. Manaus, Amazonas, 2015.

KATO, L.; BRAGA, R.; PAZMINO, A. V. Kit didático para ensino de robótica. SIGRADI, Santa Catarina, p. 773–780, 2015.

KURNIAWAN, A. **Internet of Things Projects with ESP32: Build exciting and powerful IoT projects using the all-new Espressif ESP32**. 1. ed. Birmingham, Mumbai: Packt, 2019.

LEGO. **LEGO® MINDSTORMS® EV3**. 2021. Disponível em: <https://www.lego.com/pt-br/product/lego-mindstorms-ev3-31313>. Acesso em: 20 ago. 2021.

LIMA, C. B. de; VILLAÇA, M. V. M. **AVR e Arduino: técnicas de projeto**. 1. ed. Florianópolis, 2012.

LIN, E. K. et al. Can robots teach? Preliminary results on education robotics in special education, p. 319–321, 2006.

LUGLI, A. B.; SOBRINHO, D. G. TECNOLOGIAS WIRELESS PARA AUTOMAÇÃO INDUSTRIAL: WIRELESS_HART, BLUETOOTH, WISA, WI-FI, ZIGBEE E SP-100., 2012. Disponível em: <https://bit.ly/34dl0dW>.

MACHADO, R. M. et al. Ler, escrever e programar: atividades essenciais para o desenvolvimento cognitivo na era da informação. Nuevas Ideas en Informática Educativa, v. 12, p. 512–516, 2016.

MAKEBLOCK. **Makeblock - All Products**. 2021. Disponível em: <https://store.makeblock.com/collections/all>. Acesso em: 20 ago. 2021.

MDN. **Tecnologia Web para desenvolvedores-JavaScript-Tutoriais**. MDN Web Docs. Disponível em: <https://developer.mozilla.org/pt-BR/docs/Web/JavaScript>. Acesso em: 17 ago. 2021.

MICRODUINO. **Microduino - Produtos**. 2021. Disponível em: <https://www.microduino.com.br/>. Acesso em: 20 ago. 2021.

MODELIX. **Kit de Robótica - Ensino Médio - 1º,2º,3º Ano ou Superior**. 2021. Disponível em: <https://www.modelix.com.br/kit-robotica-ensino-medio>. Acesso em: 20 ago. 2021.

PAPERT, S. **Mindstorms: Children, Computers, And Powerful Idea**. Basic Books, Inc., Publishers, 1980.

REACTNATIVE. **Introdução React Native - Componentes principais e componentes nativos**. Disponível em: <https://reactnative.dev/docs/intro-react-native-components>. Acesso em: 18 ago. 2021.

RESNICK, M. **Let's teach kids to code**. TED, 2013. Disponível em: <https://www.ted.com/talks/mitch_resnick_let_s_teach_kids_to_code>. Acesso em: 12 mai. 2021.

ROBOCORE. 2020. https://www.robocore.net/loja/iot/esp32-wifi-bluetooth. Acesso em: 17 jun. 2021.

SILVA, F. I. da; SCHERER, D. Praxedes: Protótipo de um Kit Educacional de Robótica Baseado na Plataforma Arduino. Revista EaD - Tecnologias Digitais na Educação, 2013.

SILVA, J. C. da. Ensino de Programação para alunos do Ensino Básico: Um levantamento das pesquisas realizadas no Brasil. Paraíba, 2017.

SIQUEIRA, I. C. P. Por que precisamos de Computação na Educação Básica? In: COMPUTAÇÃO Brasil. Revista da Sociedade Brasileira de Computação (SBC), 2019.

SIQUEIRA, M. D.; VALLIM, M. B. R. Um módulo de Controle para um Kit de Robótica Educacional Baseado em Hardware Reconfigurável. Londrina, p. 15–20, 2011.

SPHERO. **Sphero BOLT Coding Robot**. 2021. Disponível em: <https://sphero.com/products/sphero-bolt>. Acesso em: 20 ago. 2021.

STEFANUTO, I.; S., J. A. M.; TORRES, C. T. Evolução das Redes Sem Fio: Comparativo Entre Wi-Fi e Bluetooth. **Caderno de Estudos Tecnológicos**, v.4, n.1, 2016. Disponível em: <http://www.fatecbauru.edu.br/ojs/index.php/CET/article/view/226>.

STELLZER, R. I. C. HTML e Javascript para vários dispositivos. Mauá, 2014.

TANENBAUM, A.S. **Organização Estruturada de Computadores**. 5. ed.: Pearson Prentice Hall, 2007.

TAPARELLI, C. H. A. A evolução tecnológica do rádio. Revista USP, p. 16–21, 2002.

TOCCI, R. J.; WIDMER, Neal S.; MOSS, Gregory L. **Sistemas digitais : princípios e aplicações**. 11. ed. São Paulo: Pearson Prentice Hall, 2011.

VALENTE, J. A. **Computadores e conhecimento: repensando a educação**. 2. ed. Campinas,SP: UNICAMP/NIED, 1998.